Université Hassan II - Casablanca

Faculté des Sciences et Techniques Mohammedia

Module Informatique I 231:

Bases de Données

Parcours BCG

2014-2015

Plan du Cours

Cours (16 h)

- **Généralités (2h)**
  - Introduction générale à l'informatique et aux bases de données
- **Démarche de construction d'une BD (6h)**
  - Conception de modèles E/A
  - Passage au modèle relationnel
- **Langage SQL(8h)**
  - LDD
  - LMD

Travaux dirigés (12 h)

- **Elaboration de Modèle s conceptuel Entité/Association (3h)**
- **Passage du modèle Entité/Association au modèle relationnel (3h)**
- **Langage de Définition des Données (LDD) (3h)**
- **Langage de Manipulation des Données (LMD)(3h)**

Travaux pratiques (24h)

- **SGBD Relationnel (6h)**
- **Langage de requêtage LDD et LMD (12h)**

CHAPITRE 1:

INTRODUCTION GENERALE AUX BASES DE DONNEES

Objectifs du Chapitre

A la fin de ce chapitre, vous pourrez :

- Enumérer les différents supports de stockages et les limites d'utilisation des fichiers.
- Définir une Base de données et identifier son importance.
- Définir le rôle d'un Système de gestion de base de données relationnel et énumérer ses fonctionnalités.
- Décrire les aspects physiques et les aspects théoriques des bases de données.

Sommaire

- Motivation
- Les secteurs d’utilisation des bases de données
- Les supports de Stockage des données
- Limites du stockage dans les fichiers
- Historique
- Définition d’une Base de données et de SGBD
- Rôle d’un SGBD
- Modèles de données
- Composants et fonctionnalités d’un SGBDR
- Vocabulaire de BD
- Aperçu sur le Langage SQL
- Conclusion

Motivation (1)

- Qu’est-ce donc qu’une base de données ?
- Que peut-on attendre d’un système de gestion de bases de données ?
- Que peut-on faire avec une base de données ?
- Comment interroger et manipuler les données au sein d’une base de données?

Motivation(2)

Des données ? Est ce important pour nous ?

- Des relevés de banques, de cartes de crédit
- Des carnets d’adresses
- La consommation de téléphone
- Des inscriptions à des clubs, associations,
- Des papiers utiles
- Des horaires et disponibilités de transport
- Des programmes de télé

Secteurs d'utilisation des Bases de données(1)

Les bases de données sont omniprésentes :

- ❖ **Particulier:**
  - *Carnet d'adresse*
    - *nom, prénom, tél, email, adresse…*
- ❖ **Écoles, Universités :**
  - *Données sur les étudiants*
    - *Id, nom, prénom, classe, section, cycle, année…*
  - *Données sur les formations*
    - *matière, intervenant (enseignant), masse horaire, salle*
  - *Données sur les résultats*
    - *matière, intervenant (enseignant), pondération, résultat*
- ❖ **Entreprises**
  - *fichiers clients, fournisseurs, commandes*
  - *facturation,*
  - *gestion de stock, inventaire.*
- ❖ **Gouvernement**
  - *Données sur les citoyens*
    - *CIN, nom, prénom, date de naissance, lieu de naissance, adresse, photo…*
  - *Données sur les impôts*
    - *Raison sociale, nom, montant de taxe, date d'application…*
- ❖ …

C'est important pour vous…
C'est impératif pour les entreprises !

Secteurs d'utilisation des Bases de données(2)

Votre recherche rapide

train | vol | hotel | voiture

Composez votre voyage Alacarte® !

- Train seul
- Train & Hôtel
- Train & Voiture
- Train & Hôtel & Voiture

Au départ de	Départ (JJ/MM/AAAA)	à partir de
paris	28/09/2006	12h
A destination de	**Retour (JJ/MM/AAAA)**	**à partir de**
londres	29/09/2006	15h

Adulte 1 — 1e classe / 2e classe — 12-25, Senior, Famille, Abonnés, S'Miles...

Réserver votre billet — Consulter les horaires

France | train | vol | hôtel | voiture | loisirs

Recherche > Résultats

Il n'y a pas d'autre trajet direct ou avec une correspondance, ... afficher les trajets à 2 correspondances.

ALLERS disponibles pour le Samedi 30/09

Gares de départ et d'arrivée	Horaires	Train(s)	Durée
PARIS NORD / LONDON WATERLOO INT	18h18 / 19h58	9051	Durée : 02h42 — Choisir cet aller
PARIS NORD / LONDON WATERLOO INT	18h18 / 20h54	9055	Durée : 02h36 — Choisir cet aller
PARIS NORD / LONDON WATERLOO INT	20h43 / 22h28	9061	Durée : 02h45 — Choisir cet aller

Voir les trains suivants

Horaire des trains et réservations

3 Vos informations bancaires pour le paiement en ligne



Pour commander en toute confiance, les engagements de voyages-sncf.com sont : En savoir

- des pages sécurisées pour payer l'esprit tranquille
- l'envoi d'un email de confirmation récapitulant toutes les informations de votre commande
- un service client à votre disposition pour toutes questions à la suite de votre commande

Les symboles indiquent que votre transaction est sécurisée, vous pouvez remplir votre formulaire en toute confiance.

► Sélectionnez la carte de paiement de votre choix:



Vous pouvez utilis... solution e-Carte Bleue pour payer votre comman... En savoir

► Saisissez votre numéro de carte bancaire:

Identifiant commerçant: 055204944722232

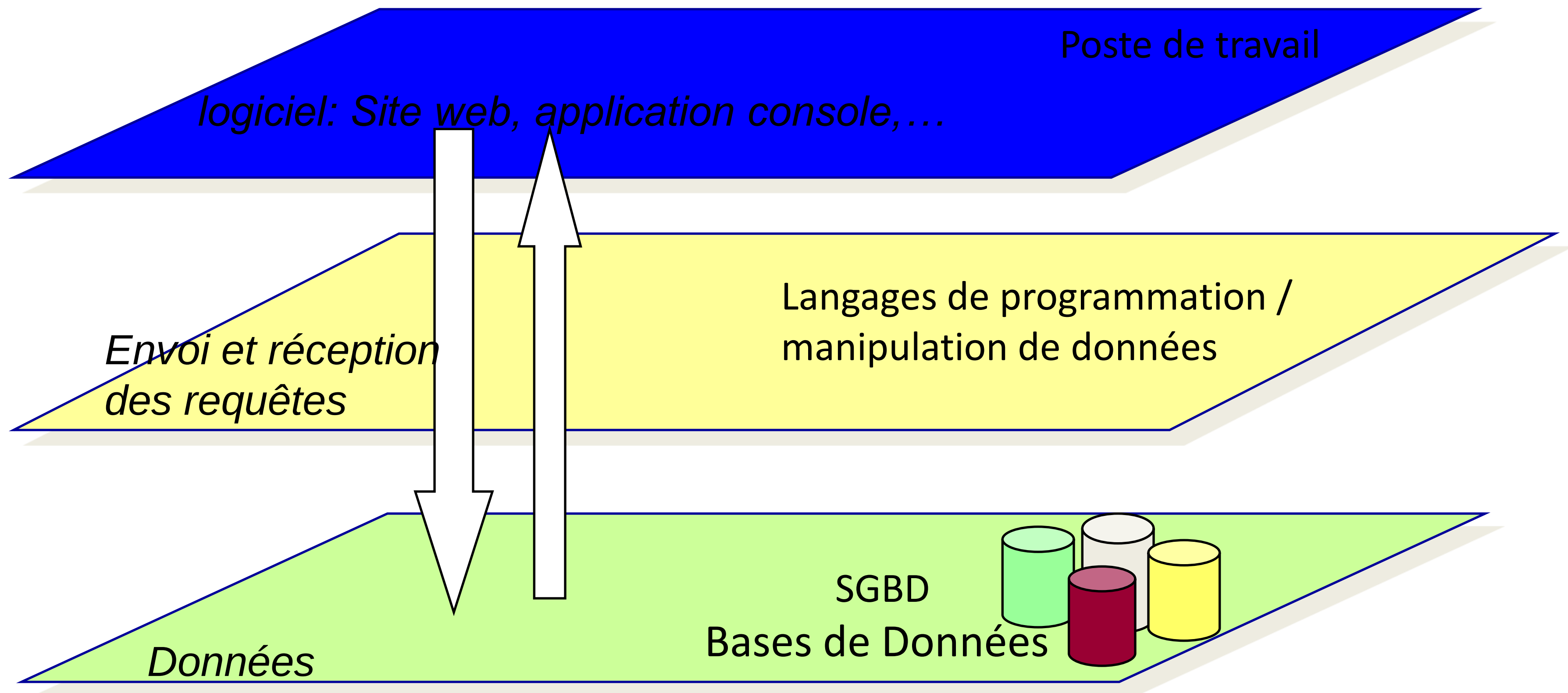
N° de carte : Expire fin : 01-Janvier / 2006

Code de sécurité :

Confirmer et payer votre commande

E-commerce
Données relatives au Paiement éléctronique

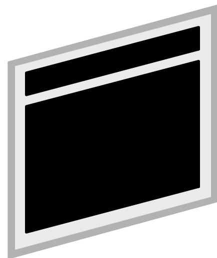
Les Bases de données dans les applications Informatiques



Les supports de Stockage de données

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	
20	Marketing	201	
50	Shipping	124	
60	IT	103	
80	Sales	149	
90	Executive	100	
110	Accounting	205	
190	Contracting		

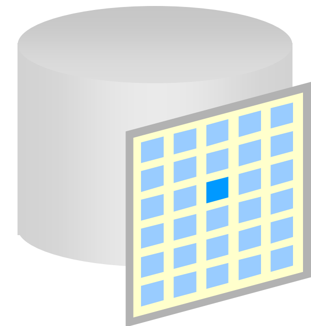
GRA	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000



Feuille de calcul électronique



Armoire de rangement



Base de données

Les limites de stockage dans les fichiers (1)

- Redondance des données et incohérences
- Isolation des données et accessibilité
- Un accès aux données = un programme
- Atomicité et environnement multi utilisateurs
- Sécurité et protection des données

Les limites de stockage dans les fichiers (2)

- Source des difficultés avec les fichiers
  - Le modèle des données est intégré dans les programmes
  - Absence de contrôle pour l'accès et la manipulation des données

Historique

- **1950-1960**
  - Des fichiers séquentiels, du ‘batch’
- **1960 – 1970**
  - Le début des bases de données hiérarchiques (ex : IMS, Information Management Systems)
  - BD réseaux ou CODASYL (Committee on Data Systems and Languages; ex : IDS, Integrated Data Store)
- **1970 – 1980**
  - La naissance du modèle relationnel (E.F. Codd, 1970)
- **Début des années 90**
  - SQL
  - BD objets (ex : o2, Versant, 1990)
  - BD hybrides objets-relationnel (ex : Oracle V8 en 1998)
- **Fin des années 90**
  - Croissance du volume des données, Internet, modèle multi tiers
  - BD natives XML (ex: Tamino de Software AG, 2000)

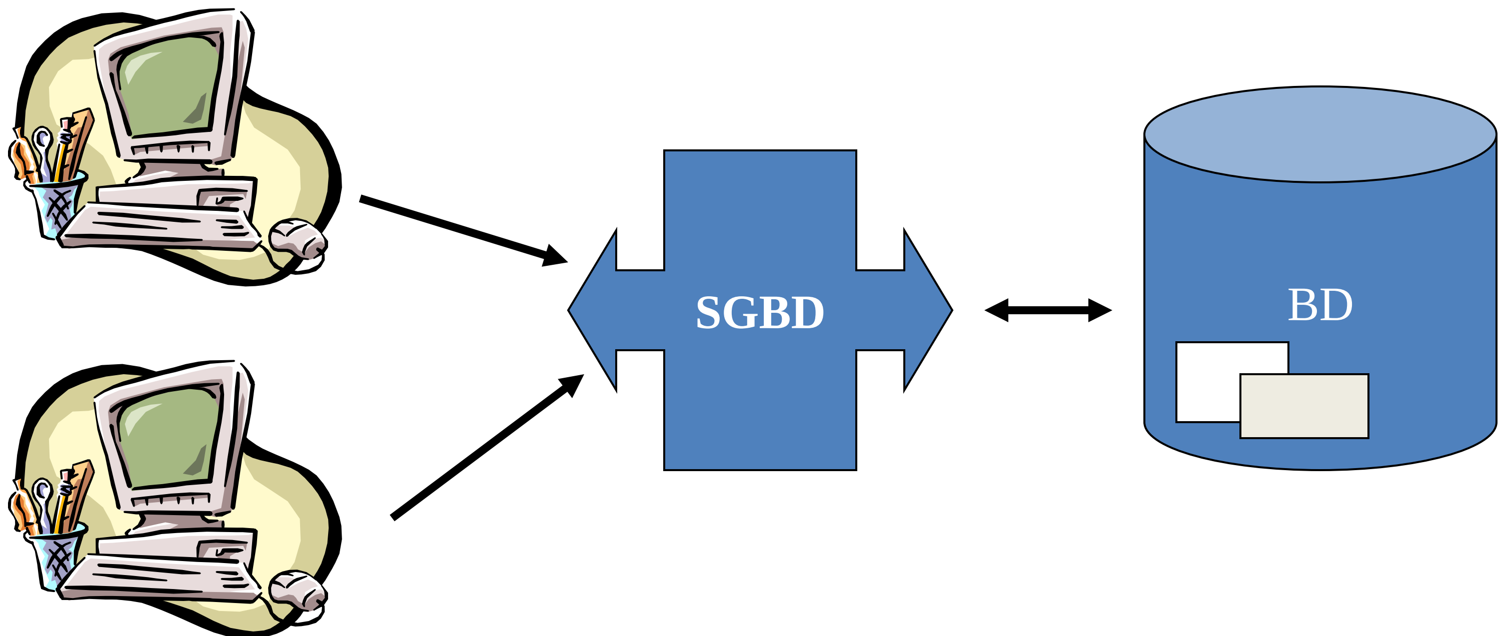
Définition d’une Base de Données

Définition: Une base de données est un ensemble structuré de données (1) enregistrées sur des supports accessibles par l’ordinateur (2) pour satisfaire simultanément plusieurs utilisateurs (3) de manière sélective (4) en un temps opportun (5).

(1) : Organisation et description de données
(2) : Stockage sur disque
(3) : Partage des données
(4) : Confidentialité
(5) : Performance

Définition d'un système de gestion de base de données

- Un système de gestion de base de données (abr. SGBD, en anglais DBMS) est un ensemble de logiciels qui sert à la manipulation des bases de données. Il sert à effectuer des opérations ordinaires telles que consulter, modifier, construire, organiser, transformer, copier, sauvegarder ou restaurer des bases de données.

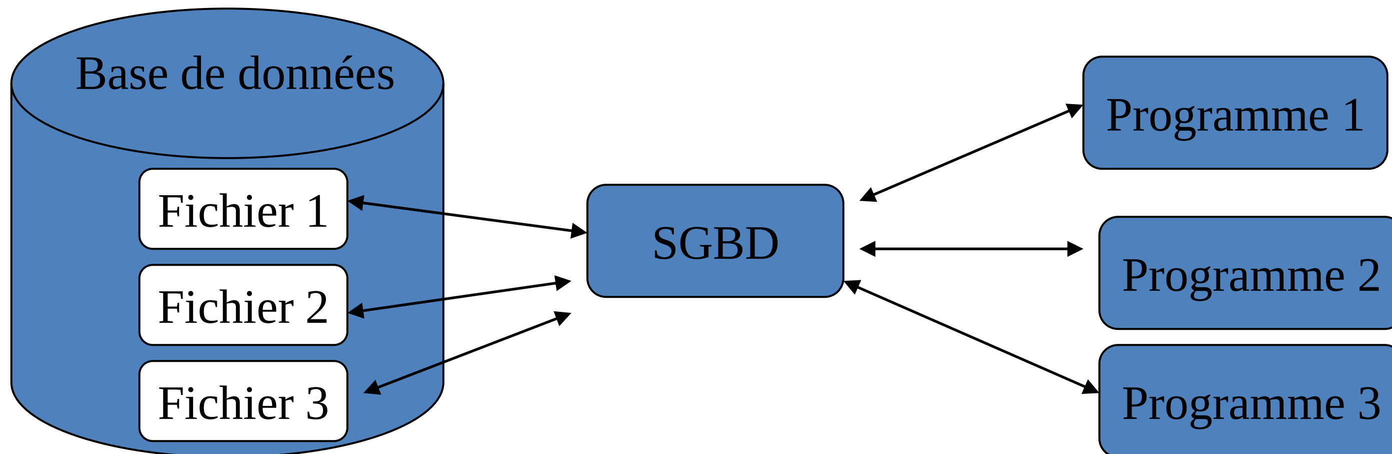


Le rôle d'un SGBD

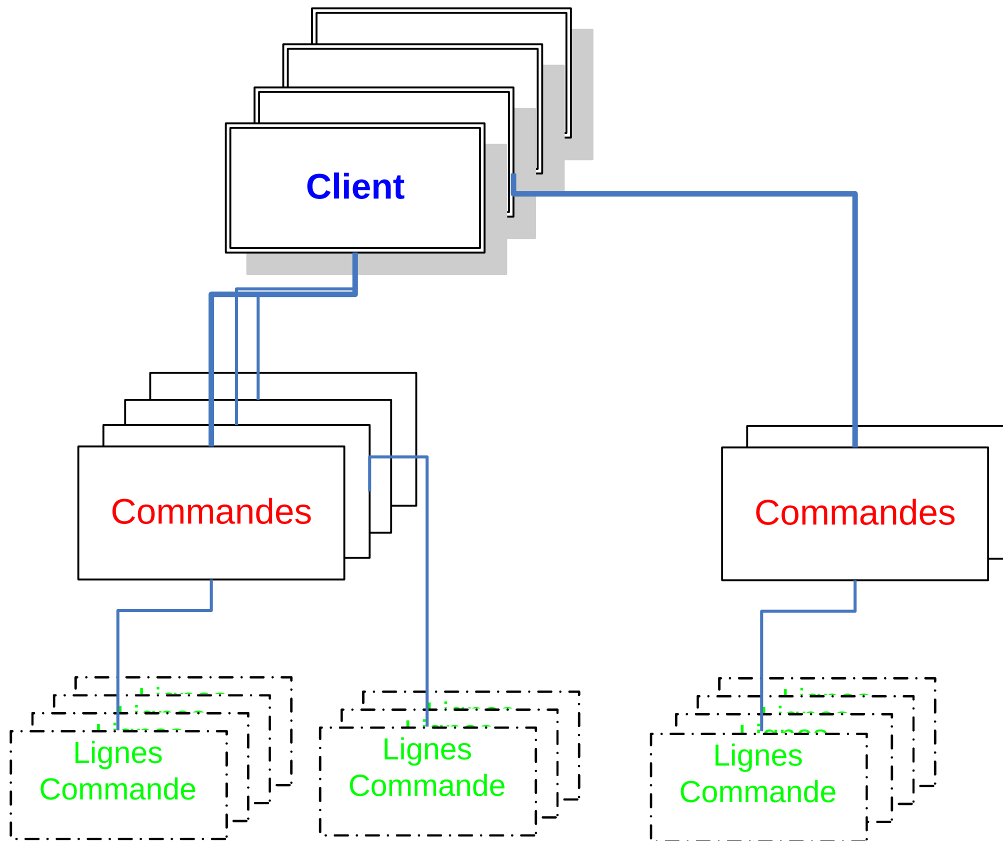
Un SGBD est un intermédiaire entre les utilisateurs et les fichiers physiques

Un SGBD facilite

- la gestion de données, avec une représentation intuitive simple sous forme de tables par exemple
- la manipulation de données. On peut insérer, modifier les données et les structures sans modifier les programmes qui manipulent la base de données



Modèle de données (1)
Modèle hiérarchique



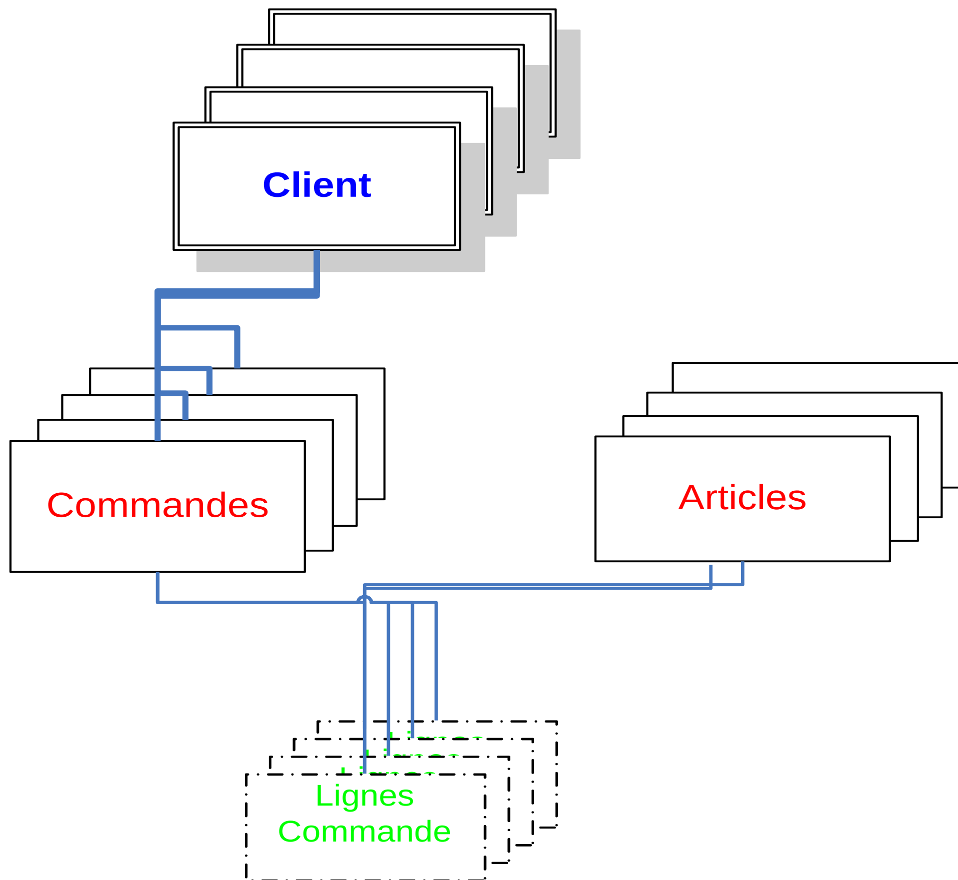
Liaison entre les objets de type 1 à n

Modèle arborescent dont le parcours se fait du père vers le fils à l'aide de pointeurs

Complexité importante

Système DL1 / IBM Années 60

Modèle de données (2)
Modèle réseau



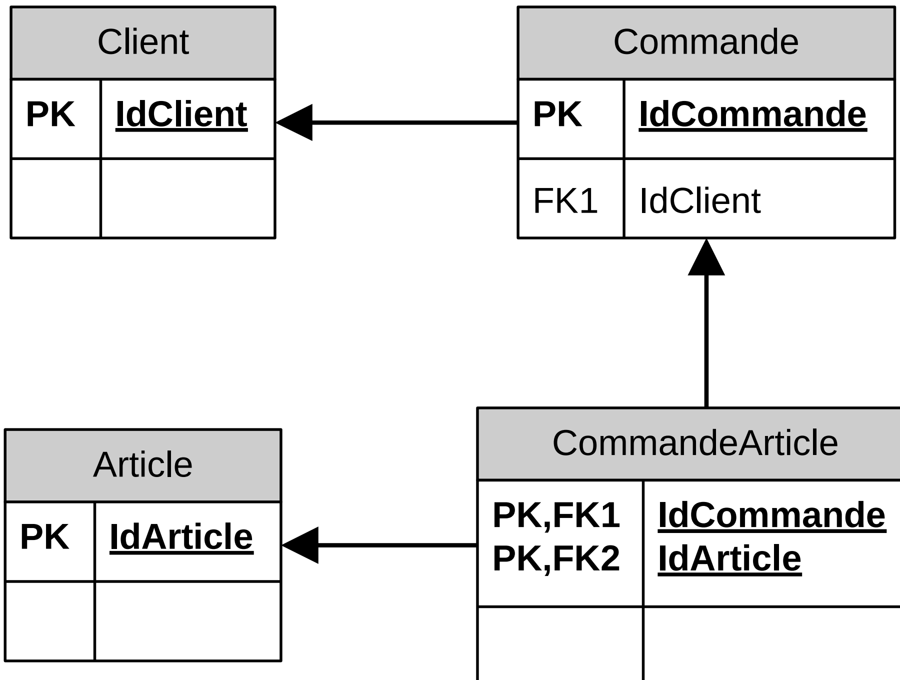
Liaison entre les objets de type n à n

Modèle dont le parcours se fait à l'aide de pointeurs mais en tous sens

Paternité multiple

Système IDS2 de Bull 1968 (Bachman 62)

Modèle de données (3)
Modèle relationnel



Basé sur le modèle Entité Relation dérivé de la théorie des ensembles et de la logique des prédicats

Grande indépendance entre vue externe et stockage interne

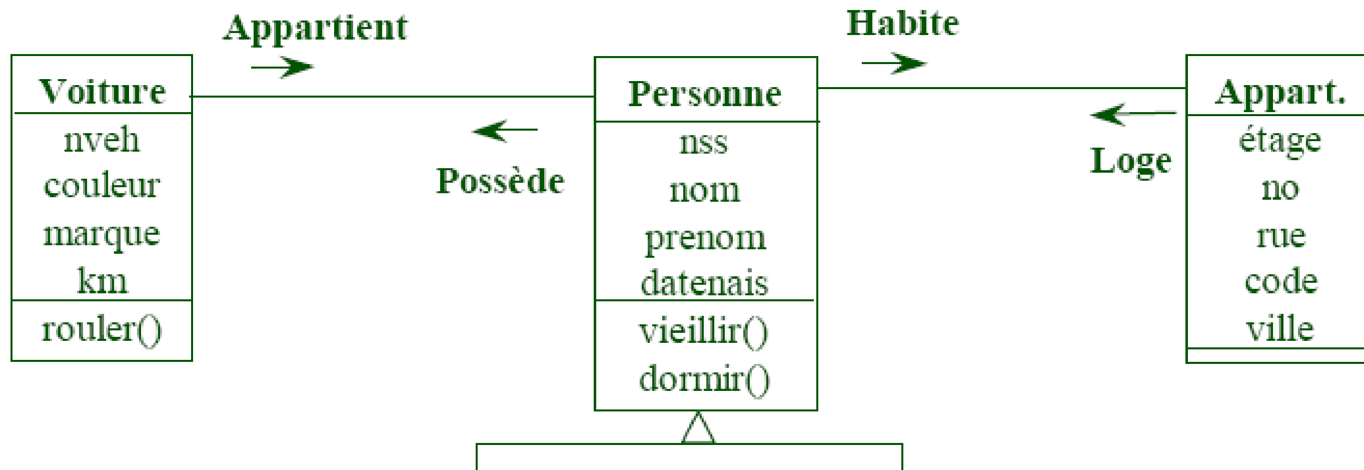
Simplicité - Evolutivité

Théorie : Codd 1972

IBM System R 1979 et DB2 1982

Modèle de données (4)

Modèle Objet



Évolution du modèle relationnel qui tendrait à simplifier les problèmes liés à la persistance et à la navigation dans les collections : Langage OQL

Langage propriétaire O2C ou intégration dans langages C, C++, Java

Exemples de SGBDR

– Dans le marché, il existe un ensemble de SGBDR très répondus et respectant le modèle relationnel :

- **DB2 - IBM**
  - **UDB (Universal Data Base Tous Systèmes)**
  - **DB2 400 (AS 400)**
- **Oracle**



- **SQL Server – Microsoft**
- **Postgres**



- **MYSQL (Open Source-Rachat par Oracle)**



- **SYBASE**





Composants d’un SGBDR

- Un SGBD est un ensemble de logiciels parmi lesquels il y a un moteur de base de données, un interprète du langage SQL, une interface de programmation, et diverses interfaces utilisateur:

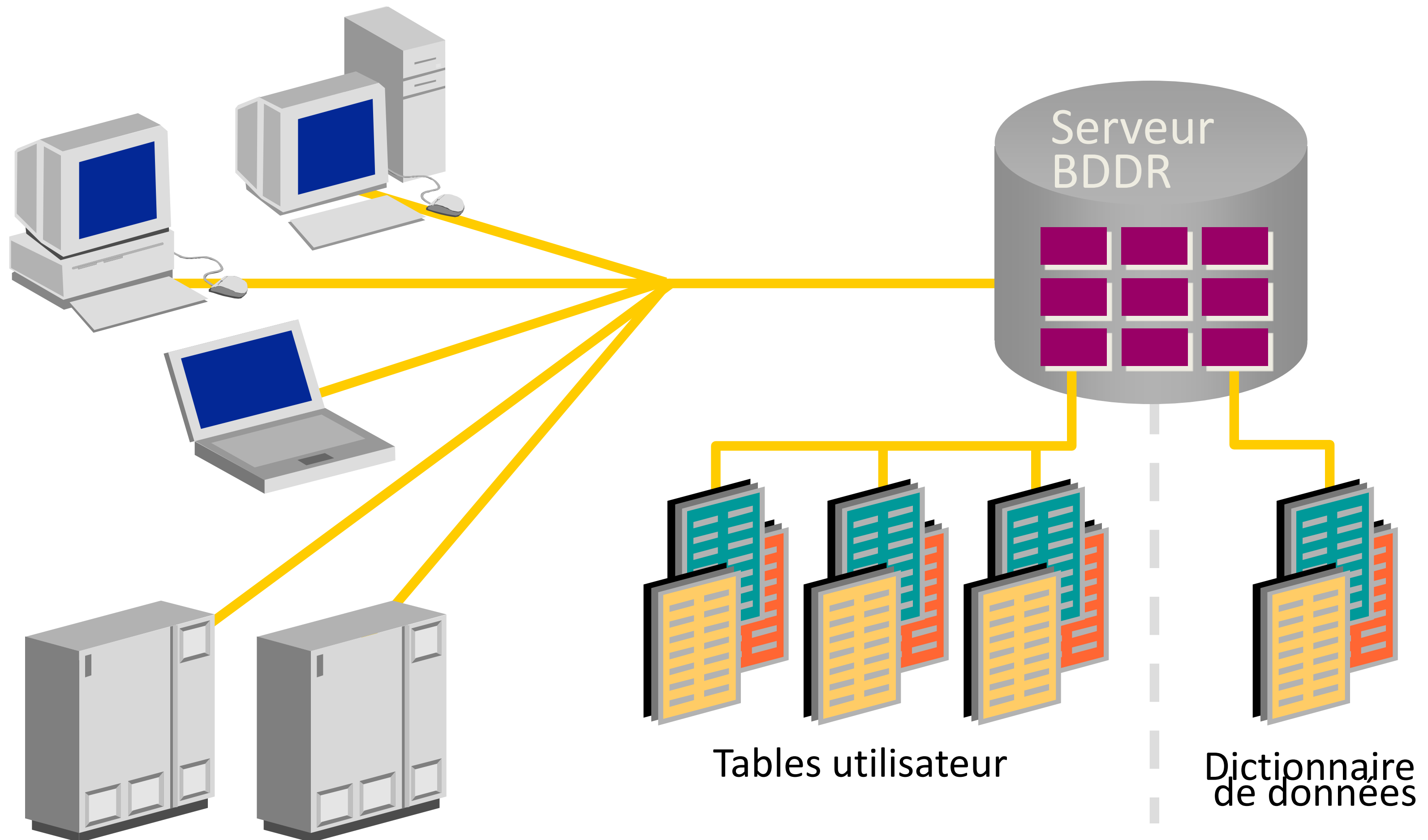
- ➢ Le moteur de base de données est le composant central du SGBD qui effectue la majorité des traitements de manipulation du contenu des bases de données.

- ➢ SQL est un langage informatique qui sert à exprimer des requêtes d'opérations sur les bases de données. L'interprète SQL décode les requêtes, et les transforme en un plan d'exécution détaillé, qui est alors transmis au moteur de base de données.

- ➢ Une interface de programmation - bibliothèque logicielle permet à un logiciel tiers de communiquer avec le SGBD, de demander des opérations et de récupérer des données provenant des bases de données. Le détail des demandes est souvent formulé en langage SQL.

- ➢ Une interface utilisateur permet aux différents catégories d’utilisateurs (administrateurs, développeurs ou utilisateurs simples), de communiquer avec le SGBD dans le but d’administrer ou manipuler les bases de données moyennant des commandes ou requêtes SQL.

Système de gestion de base de données relationnelle



Fonctionnalités d’un SGBDR (1)

- **Contrôler la redondance d'informations**
  - La redondance d'informations pose différents problèmes (coût en temps, coût en volume et risque d'incohérence entre les différentes copies). Un des objectifs des bases de données est de contrôler cette redondance, voire de la supprimer, en offrant une gestion unifiée des informations complétée par différentes vues pour des classes d'utilisateurs différents.
- **Partage des données**
  - Une base de données doit permettre d'accéder la même information par plusieurs utilisateurs en même temps. Le SGBD doit inclure un mécanisme de contrôle de la concurrence basé sur des techniques de verrouillage des données (pour éviter par exemple qu'on puisse lire une information qu'on est en train de mettre à jour).
  - Le partage des données se fait également par la notion de vue utilisateur, qui permet de définir pour chaque classe d'utilisateurs la portion de la base de données qui l'intéresse (et dans la forme qui l'intéresse).

Fonctionnalités d’un SGBDR (2)

- **Gérer les autorisations d'accès**
  - Une base de données étant multi-utilisateurs, se pose le problème de la confidentialité des données. Des droits doivent être gérés sur les données, droits de lecture, mise à jour, création, ... qui permettent d'affiner la notion de vue utilisateur.
- **Offrir des interfaces d'accès multiples**
  - Un SGBD doit offrir plusieurs interfaces d'accès, correspondant aux différents types d'utilisateurs pouvant s'adresser à lui. On trouve des interfaces orientées utilisateur final (langages de requêtes déclaratifs comme SQL avec mise en oeuvre graphique, interface de type formulaire, ...) ou bien orientées programmeurs d'applications (interface avec des langages de programmation classiques comme par exemple l'approche SQL immergé ou "embedded SQL").

Fonctionnalités d’un SGBDR (3)

- **Représenter des relations complexes entre les données**
  - Un SGBD doit permettre de représenter des données inter-reliées de manière complexe. Cette facilité s'exprime à travers le modèle de données sous-jacent au SGBD. Chaque modèle de données offre ses propres concepts pour représenter les relations. On peut citer les modèles hiérarchique, réseau (première génération de modèles), relationnel (génération actuelle), sémantiques (ou orientés vers la conception tel que Entité-Association, Z, ...) ou orienté-objet.
- **Vérifier les contraintes d'intégrité**
  - Un schéma de base de données se compose d'une description des données et de leurs relations ainsi que d'un ensemble de contraintes d'intégrité. Une contrainte d'intégrité est une propriété de l'application à modéliser qui renforce la connaissance que l'on en a. On peut classifier les contraintes d'intégrité, en contraintes structurelles (un employé a un chef et un seul par exemple) et contraintes dynamiques (un salaire ne peut diminuer). Les SGBD commerciaux supportent automatiquement un certain nombre de contraintes structurelles, mais ne prennent pas en compte les contraintes dynamiques (elles doivent être codées dans les programmes d'application).

Fonctionnalités d’un SGBDR (4)

- **Assurer la sécurité et la reprise après panne**
  - Une base de données est souvent vitale dans le fonctionnement d'une organisation, et il n'est pas tolérable qu'une panne puisse remettre en cause son fonctionnement de manière durable. Les SGBD fournissent des mécanismes pour assurer cette sécurité. Le premier mécanisme est celui de transaction qui permet d'assurer un comportement atomique à une séquence d'actions (elle s'effectue complètement avec succès ou elle est annulée). Une transaction est une séquence d'opérations qui fait passer la base de données d'un état cohérent à un nouvel état cohérent. L'exemple typique est celui du débit-crédit pour la gestion d'une carte bancaire. Ce mécanisme permet de s'affranchir des petites pannes (style coupure de courant).
  - En ce qui concerne les risques liés aux pannes disques, les SGBD s'appuie sur un mécanisme de journalisation qui permet de regénérer une base de données automatiquement à partir d'une version de sauvegarde et du journal des mouvements.

Vocabulaire de BD (1)

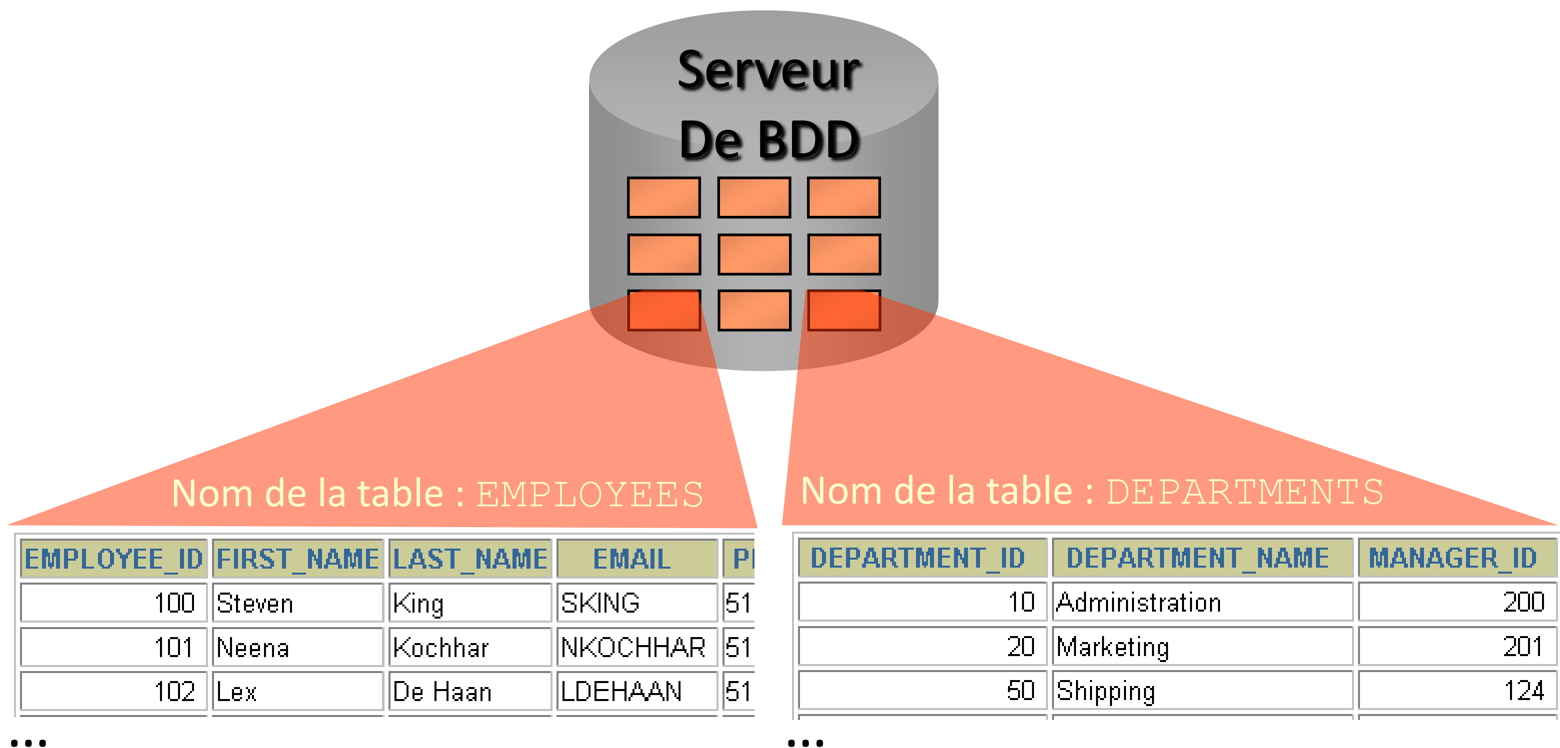
- Une base de données est composée de différents objets (Index, Séquence, Vue, Table,…). Les TABLES constituent un objet principal d'une base de données.

- Une table contient des COLONNES , qui contiennent les différentes valeurs de données relatives à un attribut.

- Les ENREGISTREMENTS correspondent aux données saisies *(= ligne).*

Vocabulaire de BD (2)

Une base de données relationnelle est un ensemble de relations ou de tables à deux dimensions.



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PI
100	Steven	King	SKING	51
101	Neena	Kochhar	NKOCHHAR	51
102	Lex	De Haan	LDEHAAN	51

…

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID
10	Administration	200
20	Marketing	201
50	Shipping	124

…

Vocabulaire de BD (3):
Table

numcli	nomcli	prenomcli	adrcli	cpcli	villecli
10003	ANDRE	Thomas	10 rue courtalon	75014	PARIS
102778	BULI	Patrice	20 cours Vuitton	69006	LYON
124682	VERIN	Didier	10 rue Schuman	75002	PARIS
125941	GADRON	Dominique	10 allée des Tilleuls	69300	CALUIRE
140621	BALOIR	Claude	10 place des cordeliers	75008	PARIS
143025	BLANC	Cécile	12 rue Cortier	89000	AUXERRE
147963	DOLMEN	Christian	145 cous Lafayette	75015	PARIS
148254	JORET	Geneviève	10 rue des provinces	69005	LYON
150763	MALOT	Berthe	39 rue denfert rochereau	75014	PARIS
157269	OLIVE	Virginie	20 rue Berliet	21000	DIJON
164025	PALO	Nicole	10 rue de la Préfecture	75001	PARIS
172436	MARIE	Olivier	43 rue de l'église	06000	NICE
172596	BELO	Vincent	15 rue des chèvrefeuilles	75002	PARIS
175298	GIDOT	Sylvie	20 rue de Melun	75013	PARIS
175496	FAVRE	Caroline	61 rue du calvaire	10000	TROYES
240573	VIAR	Stéphane	68 rue Anatole France	75016	PARIS
279246	MAIRE	René	28 rue de l'Orme	69003	LYON

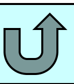
Table CLIENT

Vocabulaire de BD (4)
Colonnes

La clé primaire de la table CLIENT

Colonnes de la table CLIENT

numcli	nomcli	prenomcli	adrcli	cpcli	villecli
10003	ANDRE	Thomas	10 rue courtalon	75014	PARIS
102778	BULI	Patrice	20 cours Vuitton	69006	LYON
124682	VERIN	Didier	10 rue Schuman	75002	PARIS
125941	GADRON	Dominique	10 allée des Tilleuls	69300	CALUIRE



Vocabulaire de BD (3): les enregistrements

numcli	nomcli	prenomcli	adrcli	cpcli	villecli
10003	ANDRE	Thomas	10 rue courtalon	75014	PARIS
102778	BULI	Patrice	20 cours Vuitton	69006	LYON
124682	VERIN	Didier	10 rue Schuman	75002	PARIS
125941	GADRON	Dominique	10 allée des Tilleuls	69300	CALUIRE
140621	BALOIR	Claude	10 place des cordeliers	75008	PARIS
143025	BLANC	Cécile	12 rue Cortier	89000	AUXERRE
147963	DOLMEN	Christian	145 cous Lafayette	75015	PARIS
148254	JORET	Geneviève	10 rue des provinces	69005	LYON
150763	MALOT	Berthe	39 rue denfert rochereau	75014	PARIS
157269	OLIVE	Virginie	20 rue Berliet	21000	DIJON
164025	PALO	Nicole	10 rue de la Préfecture	75001	PARIS
172436	MARIE	Olivier	43 rue de l'église	06000	NICE
172596	BELO	Vincent	15 rue des chèvrefeuilles	75002	PARIS
175298	GIDOT	Sylvie	20 rue de Melun	75013	PARIS
175496	FAVRE	Caroline	61 rue du calvaire	10000	TROYES
240573	VIAR	Stéphane	68 rue Anatole France	75016	PARIS
279246	MAIRE	René	28 rue de l'Orme	69003	LYON

Un enregistrement de la table CLIENT

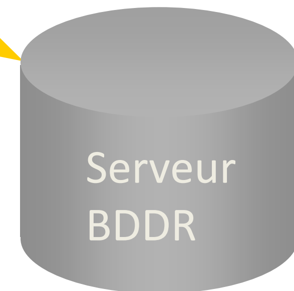
Aperçu sur le langage SQL (1)

Les SGBDR résolvent sur demande (anglais query) des calculs utilisant des opérateurs d'algèbre relationnelle. SQL (sigle de Structured Query Language) est le langage informatique le plus répandu utilisé pour formuler des calculs d'algèbre relationnelle.

L'instruction SQL est entrée.

```
SELECT department_name
FROM   departments;
```

L'instruction est envoyée au SGBDR.



DEPARTMENT_NAME
Administration
Marketing
Shipping
IT
Sales
Executive
Accounting
Contracting

La réponse est envoyée à l'utilisateur.

Aperçu sur le langage SQL(2)
Langage de définition des données

- LDD (Langage de Définition de Données)
- Permet de décrire les données
  - Type, Longueur, Nature
  - Valeurs acceptées (contraintes d'intégrité sur domaine)
  - Règles de gestion statiques (contraintes d'intégrité fonctionnelles)

Aperçu sur le langage SQL(3)
Langage de manipulation de données

- LMD (le Langage de Manipulation des Données)
- Permet de réaliser les opérations
  - Ajout, modification, suppression
  - Sélection
- Interactif ou Batch

Conclusion

- Ce chapitre est une introduction aux bases de données, qui permet d'approcher aux étudiants du BCG les notions fondamentales qui constituent une assise à la suite du module.
- Avant d'apprendre comment créer une base de données dans un SGBDR, et communiquer ensuite avec celle-ci en utilisant le langage SQL, il est nécessaire de procéder par une étape préliminaire à savoir la Conception.

Prof O. EL BEGGAR

CHAPITRE 2:
CRÉER ET GÉRER DES TABLES

Objectifs

A la fin de ce chapitre, vous pourrez :

- créer une base de données MySQL
- créer des tables
- décrire les différents types de données utilisables pour les définitions de colonne
- modifier des définitions de table
- Supprimer des tables

Objets de base de données

Objet	Description
Table	Unité de stockage élémentaire, composée de lignes et de colonnes
Vue	Représentation logique de sous-ensembles de données issus d'une ou de plusieurs tables
Séquence	Générateur de valeurs numériques
Index	Améliore les performances de certaines interrogations
Synonyme	Permet d'affecter un autre nom à un objet

Instruction CREATE Database

- Vous devez indiquer :
  - le nom de la base de données,
  - Utiliser ensuite cette base de données, pour y créer des tables.

```
CREATE DATABASE nombase;
Use nombase;
```

Instruction CREATE TABLE

- Vous devez disposer :
  - du privilège CREATE TABLE,
  - d'un espace de stockage

```
CREATE TABLE nomtable
          (nomcolumn datatype [DEFAULT expr][, ...]);
```

- Vous devez indiquer :
  - le nom de la table,
  - le nom, le type de données et la taille des colonnes.

Option DEFAULT

- Permet d'indiquer la valeur par défaut d'une colonne lors d'une insertion.

```
... commission INT DEFAULT 0, ...
```

- Valeurs autorisées : valeurs littérales, expressions et fonctions SQL.
- Valeurs non autorisées : noms d'autres colonnes ou pseudo-colonnes.
- Le type de données par défaut doit correspondre à celui de la colonne.

Créer des tables

- Créez la table.

```
CREATE TABLE dept
              (deptno  INT(2),
               dname   VARCHAR(14),
               loc     VARCHAR(13));
```

Votre requête SQL a été exécutée avec succès (Traitement en 0.0770 sec.)

- Vérifiez la création de la table.

```
DESCRIBE dept
```

Field	Type	Null	Key	Default	Extra
deptno	int(2)	YES		NULL	
dname	varchar(14)	YES		NULL	
loc	varchar(13)	YES		NULL	

Tables de la base de données MySQL

- Les tables utilisateur :
  - constituent un ensemble de tables créées et gérées par l'utilisateur,
  - contiennent des informations relatives à l'utilisateur.
- Le dictionnaire de données :
  - constitue un ensemble de tables créées et gérées par le serveur MySQL,
  - contient des informations relatives à la base de données.

Interroger le dictionnaire de données

- Consulter les tables crées par l'utilisateur.

```
Show tables;
```

- Résultat

Tables_in_BDexemple
Employé
Département
Localité

Types de données

Data Type	Description
VARCHAR(*size*)	Données alphanumériques de longueur variable
CHAR(*size*)	Données alphanumériques de longueur fixe
DECIMAL(*p*,*s*)	Données numériques de longueur variable
DATE	Valeurs de date et d'heure
FLOAT	Données numériques réelles codées sur 32 bits
DOUBLE	Données numériques réelles codées sur 64 bits
TEXT	Données textuelle
BOOLEAN	Données booléenne acceptant comme valeur vrai/faux
INT(size)	Données entière de longueur variable

Instruction ALTER TABLE

L'instruction ALTER TABLE permet :

- d'ajouter une nouvelle colonne,
- de modifier une colonne existante,
- de définir une valeur par défaut pour une nouvelle colonne,
- de supprimer une colonne.

Instruction ALTER TABLE

L'instruction ALTER TABLE permet d'ajouter, de modifier ou de supprimer des colonnes.

```
ALTER TABLE table
ADD         (column datatype [DEFAULT expr]
            [, column datatype]...);
```

```
ALTER TABLE table
MODIFY      (column datatype [DEFAULT expr]
            [, column datatype]...);
```

```
ALTER TABLE table
DROP        (column);
```

Ajouter une colonne

Nouvelle colonne

DEPT80

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE
149	Zlotkey	126000	29-JAN-00
174	Abel	132000	11-MAY-96
176	Taylor	103200	24-MAR-98

JOB_ID

"Ajoutez une colonne à la table DEPT80."

DEPT80

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
149	Zlotkey	126000	29-JAN-00	
174	Abel	132000	11-MAY-96	
176	Taylor	103200	24-MAR-98	

Ajouter une colonne

- La clause ADD permet d'ajouter des colonnes.

```
ALTER TABLE dept80
ADD         (job_id VARCHAR(9));
Table altered.
```

- La nouvelle colonne est placée à la fin de la table.

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
149	Zlotkey	126000	29-JAN-00	
174	Abel	132000	11-MAY-96	
176	Taylor	103200	24-MAR-98	

Modifier une colonne

- Vous pouvez modifier le type de données, la taille et la valeur par défaut d'une colonne.

```
ALTER TABLE   dept80
MODIFY        (last_name VARCHAR(30));
Table altered.
```

- La modification d'une valeur par défaut ne s'applique qu'aux insertions ultérieures dans la table.

Supprimer une colonne

La clause DROP COLUMN permet de supprimer d'une table les colonnes qui ne sont plus utiles.

```
ALTER TABLE  dept80
DROP COLUMN  job_id;
Table altered.
```

Supprimer une table

- La structure et l'ensemble des données de la table sont supprimées.
- Toutes les transactions en cours sont validées.
- Tous les index sont supprimés.
- Vous ne *pouvez pas* annuler une instruction DROP TABLE.

```
DROP TABLE dept80;
Table dropped.
```

Synthèse

Ce chapitre vous a permis d'apprendre à utiliser des instructions LDD pour créer, modifier, supprimer et renommer des tables.

Instruction	Description
CREATE TABLE	Crée une table
ALTER TABLE	Modifie la structure d'une table
DROP TABLE	Supprime les lignes et la structure d'une table

Prof O. EL BEGGAR

CHAPITRE 3:
INCLURE DES CONTRAINTES

Objectifs

A la fin de ce chapitre, vous pourrez :

- définir des contraintes d’intégrité
- créer et gérer des contraintes d’intégrité

Qu'est-ce qu'une contrainte ?

- Les contraintes appliquent des règles au niveau d'une table.
- Les contraintes empêchent la suppression d'une table lorsqu'il existe des dépendances.
- Les types de contrainte suivants sont utilisés :
  - `NOT NULL`
  - `UNIQUE`
  - `PRIMARY KEY`
  - `FOREIGN KEY`
  - `CHECK`

Règles applicables aux contraintes

- Vous pouvez affecter un nom aux contraintes ou laisser le serveur MySQL en générer un.
- Vous pouvez créer une contrainte :
  - au moment de la création de la table,
  - une fois que la table est créée.
- Définissez une contrainte au niveau table ou colonne.

Définir des contraintes


```
CREATE TABLE nomtable
             (column datatype [DEFAULT expr]
              [column_constraint],
              ...
              [table_constraint][,...]);
```

```
CREATE TABLE employees(
              employee_id  INT(6),
              first_name   VARCHAR(20),
              ...
              job_id       VARCHAR(10) NOT NULL,
              CONSTRAINT emp_emp_id_pk
                           PRIMARY KEY (EMPLOYEE_ID));
```

Définir des contraintes

- Contrainte au niveau colonne

```
Column constraint_type,
```

- Contrainte au niveau table

```
column,...
  [CONSTRAINT constraint_name] constraint_type
  (column, ...),
```

Contrainte NOT NULL

Interdit les valeurs NULL dans la colonne :

EMPLOYEE_ID	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID
100	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000	90
101	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000	90
102	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000	90
103	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000	60
104	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000	60
178	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	7000	
200	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	4400	10

...

20 rows selected.

Contrainte NOT NULL (aucune ligne de cette colonne ne peut contenir de valeur NULL)

Contrainte NOT NULL

Absence de contrainte NOT NULL (les lignes de cette colonne peuvent contenir une valeur NULL)

Contrainte NOT NULL

Cette contrainte est définie au niveau colonne :

```
CREATE TABLE employees(
    employee_id    INT(6) Primary key,
    last_name      VARCHAR(25) NOT NULL,
    salary         DECIMAL(8,2),
    commission_pct DECIMAL(2,2),
    hire_date      DATE,
                   CONSTRAINT emp_pk
                   NOT NULL(employee_id),
...
```

Nom attribué par le système

Nom attribué par l'utilisateur

Contrainte UNIQUE

Contrainte UNIQUE

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	EMAIL
100	King	SKING
101	Kochhar	NKOCHHAR
102	De Haan	LDEHAAN
103	Hunold	AHUNOLD
104	Ernst	BERNST

...

INSERT INTO

208	Smith	JSMITH	**Autorisé**
209	Smith	JSMITH	**Non autorisé : existe déjà**

Contrainte UNIQUE

Cette contrainte est définie au niveau table ou colonne :

```
CREATE TABLE employees(
    employee_id       INT(6),
    last_name         VARCHAR(25) NOT NULL,
    email             VARCHAR(25),
    salary            DECIMAL(8,2),
    commission_pct    DECIMAL(2,2),
    hire_date         DATE NOT NULL,
...
    CONSTRAINT emp_email_uk UNIQUE(email));
```

Contrainte PRIMARY KEY

DEPARTMENTS

PRIMARY KEY

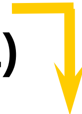
DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500

...



INSERT INTO

Non autorisé (valeur NULL)



	Public Accounting		1400
50	Finance	124	1500



Non autorisé (50 existe déjà)

Contrainte PRIMARY KEY

Cette contrainte est définie au niveau table ou colonne :

```
CREATE TABLE   departments(
    department_id          INT(4),
    department_name        VARCHAR(30)
      CONSTRAINT dept_name_nn NOT NULL,
    manager_id             INT(6),
    location_id            INT(4),
      CONSTRAINT dept_id_pk PRIMARY KEY(department_id));
```

Contrainte FOREIGN KEY

DEPARTMENTS

PRIMARY KEY



DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500

...



EMPLOYEES



FOREIGN KEY

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
102	De Haan	90
103	Hunold	60
104	Ernst	60
107	Lorentz	60

...



INSERT INTO

200	Ford	9
201	Ford	60



Non autorisé (9 n'existe pas)



Autorisé

Contrainte FOREIGN KEY

Cette contrainte est définie au niveau table ou colonne :

```
CREATE TABLE employees(
    employee_id      INT(6),
    last_name        VARCHAR(25) NOT NULL,
    email            VARCHAR(25),
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
...
    department_id    NUMBER(4),
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
      REFERENCES departments(department_id),
    CONSTRAINT emp_email_uk UNIQUE(email));
```

Mots-clés associés à la contrainte FOREIGN KEY

- FOREIGN KEY : définit une colonne de la table enfant au niveau table.
- REFERENCES : identifie la table et la colonne dans la table parent.
- ON DELETE CASCADE : supprime les lignes dépendantes de la table enfant lorsqu'une ligne de la table parent est supprimée.

Contrainte CHECK

- Définit une condition que chaque ligne doit satisfaire.

Au niveau colonne

```
..., salary   DECIMAL(8,2) CHECK (salary > 0),...
```

Au niveau Table

```
..., salary   DECIMAL(8,2),
     CONSTRAINT emp_salary_min
            CHECK (salary > 0),...
```

Ajouter une syntaxe de contrainte

Utilisez l'instruction ALTER TABLE pour :

- ajouter ou supprimer une contrainte sans modifier sa structure,
- ajouter une contrainte NOT NULL à l'aide de la clause MODIFY.

```
ALTER TABLE table
ADD [CONSTRAINT constraint] type (column);
```

Ajouter une contrainte

Ajoutez à la table EMPLOYEES une contrainte FOREIGN KEY précisant qu'un manager doit déjà exister dans cette table en tant qu'employé valide.

```
ALTER TABLE     employees
ADD CONSTRAINT  emp_manager_fk
  FOREIGN KEY(manager_id)
  REFERENCES employees(employee_id);
Table altered.
```

Supprimer une contrainte

- Supprimez de la table EMPLOYEES la contrainte relative au manager.

```
ALTER TABLE       employees
DROP CONSTRAINT   emp_manager_fk;
Table altered.
```

Synthèse

Ce chapitre vous à permis d'apprendre à créer des
contraintes.

- Types de contrainte :
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK

Prof O. EL BEGGAR

CHAPITRE 4:
ECRIRE DES INSTRUCTIONS SQL SELECT ÉLÉMENTAIRES

Objectifs

A la fin de ce chapitre, vous pourrez :

- énumérer toutes les possibilités offertes par les instructions SQL `SELECT`
- exécuter une instruction `SELECT` élémentaire

Différentes fonctions des instructions SQL SELECT

Projection

Sélection

Table 1

Table 1

Jointure



Table 1

Table 2

Instruction SELECT élémentaire


```
SELECT   *|{[DISTINCT] column|expression [alias],...}
FROM     table;
```

- SELECT indique *quelles* colonnes renvoyer
- FROM indique *dans quelle* table rechercher

Sélectionner toutes les colonnes


```
SELECT *
FROM   departments;
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

Sélectionner des colonnes spécifiques


```
SELECT department_id, location_id
FROM   departments;
```

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

Ecrire des instructions SQL

- Les instructions SQL peuvent être écrites indifféremment en majuscules et/ou minuscules.
- Les instructions SQL peuvent être écrites sur une ou plusieurs lignes.
- Les mots-clés ne doivent pas être abrégés, ni scindés sur plusieurs lignes.

Expressions arithmétiques

Créez des expressions contenant des données de type NUMBER et DATE à l'aide d'opérateurs arithmétiques.

Opérateur	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division

Utiliser des opérateurs arithmétiques


```
SELECT last_name, salary, salary + 300
FROM   employees;
```

LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernst	6000	6300

...

Hartstein	13000	13300
Fay	6000	6300
Higgins	12000	12300
Gietz	8300	8600

20 rows selected.

Priorité des opérateurs



- La multiplication et la division ont priorité sur l'addition et la soustraction.
- Les opérateurs de niveau de priorité identique sont évalués de gauche à droite.
- Les parenthèses permettent de forcer la priorité d'évaluation et de clarifier les instructions.

Priorité des opérateurs


```
SELECT last_name, salary, 12*salary+100
FROM   employees;
```

LAST_NAME	SALARY	12*SALARY+100
King	24000	288100
Kochhar	17000	204100
De Haan	17000	204100
Hunold	9000	108100
Ernst	6000	72100
...		
Hartstein	13000	156100
Fay	6000	72100
Higgins	12000	144100
Gietz	8300	99700

20 rows selected.

Utiliser des parenthèses


```
SELECT last_name, salary, 12*(salary+100)
FROM   employees;
```

LAST_NAME	SALARY	12*(SALARY+100)
King	24000	289200
Kochhar	17000	205200
De Haan	17000	205200
Hunold	9000	109200
Ernst	6000	73200

...

Hartstein	13000	157200
Fay	6000	73200
Higgins	12000	145200
Gietz	8300	100800

20 rows selected.

Définir une valeur NULL

- Une valeur NULL est une valeur non disponible, non affectée, inconnue ou inapplicable.
- La valeur NULL est différente du zéro ou de l'espace.

```
SELECT last_name, job_id, salary, commission_pct
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
King	AD_PRES	24000	
Kochhar	AD_VP	17000	
...			
Zlotkey	SA_MAN	10500	.2
Abel	SA_REP	11000	.3
Taylor	SA_REP	8600	.2
...			
Gietz	AC_ACCOUNT	8300	

20 rows selected.

Valeurs NULL dans les expressions arithmétiques

Les expressions arithmétiques comportant une valeur NULL ont pour résultat une valeur NULL.

```
SELECT last_name, 12*salary*commission_pct
FROM   employees;
```

…

Zlotkey	25200
Abel	39600
Taylor	20640

…

Gietz	

20 rows selected.

Définir un alias de colonne

L'alias de colonne :

- renomme un en-tête de colonne,
- est utile dans les calculs,
- suit le nom de la colonne (le mot-clé `AS` facultatif peut être placé entre le nom de la colonne et l'alias),
- doit obligatoirement être placé entre guillemets s'il contient des espaces ou des caractères spéciaux, ou bien si les majuscules/minuscules doivent être respectées.

Utiliser des alias de colonne


```
SELECT last_name AS name, commission_pct comm
FROM   employees;
```

NAME	COMM
King	
Kochhar	
De Haan	

...

20 rows selected.

```
SELECT last_name "Name", salary*12 "Annual Salary"
FROM   employees;
```

Name	Annual Salary
King	288000
Kochhar	204000
De Haan	204000

...

20 rows selected.

Fonction de concaténation

Une fonction de concaténation :

- concatène des colonnes ou des chaînes de caractères avec d'autres colonnes,
- est représenté par la fonction (Concat),
- crée une colonne qui contient une expression alphanumérique.

Utiliser l'opérateur de concaténation


```
SELECT    Concat(last_name,job_id) AS "Employees"
FROM      employees;
```

Employees
KingAD_PRES
KochharAD_VP
De HaanAD_VP
HunoldIT_PROG
ErnstIT_PROG
LorentzIT_PROG
MourgosST_MAN
RajsST_CLERK

...

20 rows selected.

Chaînes de caractères littérales

- Un littéral est une chaîne de caractères, un nombre ou une date inclus dans la liste `SELECT`.
- Les valeurs des littéraux alphanumériques et de type date doivent être placées entre apostrophes.
- La chaîne de caractères définie apparaît sur chaque ligne renvoyée.

Utiliser des chaînes de caractères littérales


```
SELECT Concat(last_name,' is a ',job_id)
       AS "Employee Details"
FROM   employees;
```

Employee Details
King is a AD_PRES
Kochhar is a AD_VP
De Haan is a AD_VP
Hunold is a IT_PROG
Ernst is a IT_PROG
Lorentz is a IT_PROG
Mourgos is a ST_MAN
Rajs is a ST_CLERK

...

20 rows selected.

Doublons

Par défaut, le résultat d'une interrogation affiche toutes les lignes, y compris les doublons.

```
SELECT department_id
FROM   employees;
```

DEPARTMENT_ID
90
90
90
60
60
60
50
50
50

...

20 rows selected.

Eliminer les doublons

Pour éliminer les doublons, ajoutez le mot-clé `DISTINCT` dans la clause `SELECT`.

```
SELECT DISTINCT department_id
FROM   employees;
```

DEPARTMENT_ID
10
20
50
60
80
90
110

8 rows selected.

Synthèse

Ce chapitre vous à permis d'apprendre à :

- **écrire une instruction `SELECT` qui :**
  - **renvoie toutes les lignes et colonnes d'une table**
  - **renvoie certaines colonnes d'une table**
  - **utilise des alias de colonne en guise d'en-têtes de colonne descriptifs**

```
SELECT   *|{[DISTINCT] column|expression [alias],...}
FROM     table;
```

Prof O. EL BEGGAR

CHAPITRE 5:
LIMITER ET TRIER DES DONNÉES

Objectifs

A la fin de ce chapitre, vous pourrez :

- limiter le nombre de lignes extraites par une interrogation
- trier les lignes extraites par une interrogation

Limiter le nombre de lignes à l'aide d'une sélection

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90
103	Hunold	IT_PROG	60
104	Ernst	IT_PROG	60
107	Lorentz	IT_PROG	60
124	Mourgos	ST_MAN	50

…

20 rows selected.

"Extraire tous les employés du service 90"

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90

Limiter le nombre de lignes sélectionnées

- Limitez le nombre de lignes renvoyées à l'aide de la clause WHERE.

```
SELECT    *|{[DISTINCT] column|expression [alias],...}
FROM      table
[WHERE    condition(s)];
```

- La clause WHERE se place après la clause FROM.

Utiliser la clause WHERE


```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90 ;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90

Chaînes de caractères et dates

- Les chaînes de caractères et les dates doivent être placées entre apostrophes.
- La recherche tient compte des majuscules/minuscules pour les chaînes de caractères et du format pour les dates.
- Le format de date par défaut est DD-MON-RR.

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'Whalen';
```

Conditions de comparaison

Opérateur	Signification
=	Egal à
>	Supérieur à
>=	Supérieur ou égal à
<	Inférieur à
<=	Inférieur ou égal à
<>	Différent de

Utiliser des conditions de comparaison


```
SELECT last_name, salary
FROM   employees
WHERE  salary <= 3000;
```

LAST_NAME	SALARY
Matos	2600
Vargas	2500

Autres conditions de comparaison

Opérateur	Signification
`BETWEEN ...AND...`	Compris entre ... et ... (bornes comprises)
`IN(set)`	Correspond à une valeur de la liste
`LIKE`	Ressemblance partielle de chaînes de caractères
`IS NULL`	Correspond à une valeur NULL

Utiliser la condition BETWEEN

Utilisez la condition BETWEEN pour afficher des lignes en fonction d'une plage de valeurs.

```
SELECT last_name, salary
FROM   employees
WHERE  salary BETWEEN 2500 AND 3500;
```

Limite inférieure   Limite supérieure

LAST_NAME	SALARY
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500

Utiliser la condition IN

Utilisez la condition d'appartenance IN pour vérifier la présence de valeurs dans une liste.

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201);
```

EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
202	Fay	6000	201
200	Whalen	4400	101
205	Higgins	12000	101
101	Kochhar	17000	100
102	De Haan	17000	100
124	Mourgos	5800	100
149	Zlotkey	10500	100
201	Hartstein	13000	100

8 rows selected.

Utiliser la condition LIKE

- Utilisez la condition LIKE pour rechercher des chaînes de caractères valides à l'aide de caractères génériques.
- Les conditions de recherche peuvent contenir des caractères ou des nombres littéraux :
  - % représente zéro ou plusieurs caractères.
  - _ représente un caractère.

```
SELECT   first_name
FROM     employees
WHERE    first_name LIKE 'S%';
```

Utiliser la condition LIKE

- Vous pouvez combiner plusieurs caractères génériques de recherche.

```
SELECT last_name
FROM   employees
WHERE  last_name LIKE '_o%';
```

LAST_NAME
Kochhar
Lorentz
Mourgos

Utiliser les conditions NULL

Recherchez des valeurs NULL avec l'opérateur
IS NULL.

```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL;
```

LAST_NAME	MANAGER_ID
King	

Conditions logiques

Opérateur	Signification
`AND`	Renvoie `TRUE` si les *deux* conditions sont vraies
`OR`	Renvoie `TRUE` si *l'une* des conditions est vraie
`NOT`	Renvoie la valeur `TRUE` si la condition qui suit l'opérateur est fausse

Utiliser l'opérateur AND

L'opérateur AND exige que les deux conditions soient vraies.

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >=10000
AND    job_id LIKE '%MAN%';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
149	Zlotkey	SA_MAN	10500
201	Hartstein	MK_MAN	13000

Utiliser l'opérateur OR

L'opérateur OR exige que l'une des conditions soit vraie.

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR     job_id LIKE '%MAN%';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
100	King	AD_PRES	24000
101	Kochhar	AD_VP	17000
102	De Haan	AD_VP	17000
124	Mourgos	ST_MAN	5800
149	Zlotkey	SA_MAN	10500
174	Abel	SA_REP	11000
201	Hartstein	MK_MAN	13000
205	Higgins	AC_MGR	12000

8 rows selected.

Utiliser l'opérateur NOT


```
SELECT last_name, job_id
FROM   employees
WHERE  job_id
       NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP');
```

LAST_NAME	JOB_ID
King	AD_PRES
Kochhar	AD_VP
De Haan	AD_VP
Mourgos	ST_MAN
Zlotkey	SA_MAN
Whalen	AD_ASST
Hartstein	MK_MAN
Fay	MK_REP
Higgins	AC_MGR
Gietz	AC_ACCOUNT

10 rows selected.

Règles de priorité

Ordre d'évaluation	Opérateur
1	Opérateurs arithmétiques
2	Opérateur de concaténation
3	Conditions de comparaison
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Condition logique NOT
7	Condition logique AND
8	Condition logique OR

Les parenthèses permettent de modifier les règles de priorité.

Règles de priorité


```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = 'SA_REP'
OR     job_id = 'AD_PRES'
AND    salary > 15000;
```

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000
Abel	SA_REP	11000
Taylor	SA_REP	8600
Grant	SA_REP	7000

Règles de priorité

Utilisez des parenthèses pour forcer la priorité.

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  (job_id = 'SA_REP'
OR     job_id = 'AD_PRES')
AND    salary > 15000;
```

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000

Clause ORDER BY

- Triez des lignes à l'aide de la clause ORDER BY.
  - ASC : ordre croissant (par défaut)
  - DESC : ordre décroissant
- La clause ORDER BY se place à la fin de l'instruction SELECT.

```
SELECT    last_name, job_id, department_id, hire_date
FROM      employees
ORDER BY hire_date ;
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
King	AD_PRES	90	17-JUN-87
Whalen	AD_ASST	10	17-SEP-87
Kochhar	AD_VP	90	21-SEP-89
Hunold	IT_PROG	60	03-JAN-90
Ernst	IT_PROG	60	21-MAY-91

...

20 rows selected.

Trier par ordre décroissant


```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY hire_date DESC ;
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
Zlotkey	SA_MAN	80	29-JAN-00
Mourgos	ST_MAN	50	16-NOV-99
Grant	SA_REP		24-MAY-99
Lorentz	IT_PROG	60	07-FEB-99
Vargas	ST_CLERK	50	09-JUL-98
Taylor	SA_REP	80	24-MAR-98
Matos	ST_CLERK	50	15-MAR-98
Fay	MK_REP	20	17-AUG-97
Davies	ST_CLERK	50	29-JAN-97

...

20 rows selected.

Trier par alias de colonne


```
SELECT employee_id, last_name, salary*12 annsal
FROM   employees
ORDER BY annsal;
```

EMPLOYEE_ID	LAST_NAME	ANNSAL
144	Vargas	30000
143	Matos	31200
142	Davies	37200
141	Rajs	42000
107	Lorentz	50400
200	Whalen	52800
124	Mourgos	69600
104	Ernst	72000
202	Fay	72000
178	Grant	84000

...

20 rows selected.

Trier sur plusieurs colonnes

- L'ordre des éléments de la liste ORDER BY donne l'ordre du tri.

```
SELECT last_name, department_id, salary
FROM   employees
ORDER BY department_id, salary DESC;
```

LAST_NAME	DEPARTMENT_ID	SALARY
Whalen	10	4400
Hartstein	20	13000
Fay	20	6000
Mourgos	50	5800
Rajs	50	3500
Davies	50	3100
Matos	50	2600
Vargas	50	2500

...

20 rows selected.

- Vous pouvez effectuer un tri sur une colonne ne figurant pas dans la liste SELECT.

Synthèse

Ce chapitre vous à permis d'apprendre à :

- **utiliser la clause `WHERE` pour limiter le nombre de lignes de résultat**
  - **utiliser les conditions de comparaison**
  - **utiliser les conditions `BETWEEN`, `IN`, `LIKE` et `NULL`**
  - **appliquer les opérateurs logiques `AND`, `OR` et `NOT`**
- **utiliser la clause `ORDER BY` pour trier les lignes de résultat**

```
SELECT     *|{[DISTINCT] column|expression [alias],...}
FROM       table
[WHERE     condition(s)]
[ORDER BY  {column, expr, alias} [ASC|DESC]];
```

Prof. Omar EL BEGGAR

AFFICHER DES DONNÉES ISSUES DE PLUSIEURS TABLES

Objectifs

A la fin de ce chapitre, vous pourrez :

- écrire des instructions `SELECT` pour accéder aux données de plusieurs tables en utilisant des équijointures

Afficher des données issues de plusieurs tables

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700



EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
200	10	Administration
201	20	Marketing
202	20	Marketing
...		
102	90	Executive
205	110	Accounting
206	110	Accounting

Produits cartésiens

- Un produit cartésien est généré :
  - lorsqu'une condition de jointure est omise,
  - lorsqu'une condition de jointure est incorrecte,
  - lorsque toutes les lignes de la première table sont jointes à toutes les lignes de la seconde.
- Pour éviter tout produit cartésien, insérez une condition de jointure correcte dans la clause `WHERE`.

Générer un produit cartésien

EMPLOYEES (20 lignes)

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

20 rows selected.

DEPARTMENTS (8 lignes)

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700

8 rows selected.

Produit cartésien : 20x8=160 lignes

EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
100	90	1700
101	90	1700
102	90	1700
103	60	1700
104	60	1700
107	60	1700
...		

160 rows selected.

Joindre des tables

Une jointure sert à interroger des données de plusieurs tables.

```
SELECT   table1.column, table2.column
FROM     table1, table2
WHERE    table1.column1 = table2.column2;
```

- Ecrivez la condition de jointure dans la clause WHERE.
- Placez le nom ou l'alias de la table avant le nom de la colonne lorsque celui-ci figure dans plusieurs tables.

Définition d'une équijointure

EMPLOYEES

EMPLOYEE_ID	DEPARTMENT_ID
200	10
201	20
202	20
124	50
141	50
142	50
143	50
144	50
103	60
104	60
107	60
149	80
174	80
176	80

...

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
20	Marketing
50	Shipping
50	Shipping
50	Shipping
50	Shipping
50	Shipping
60	IT
60	IT
60	IT
80	Sales
80	Sales
80	Sales

...





Clé étrangère **Clé primaire**

Equijointures

- Joindre Les deux tables employees et departments consiste en :
  - Pour déterminer le service d'un employé, vous devez comparer la valeur de la colonne DEPARTMENT_ID de la table EMPLOYEES à celles de la colonne DEPARTMENT_ID de la table DEPARTMENTS. La relation entre les tables EMPLOYEES et DEPARTMENTS est appelée *équijointure* puisque les valeurs de la colonne DEPARTMENT_ID des deux tables sont égales. Ce type de jointure implique fréquemment l'utilisation de clés primaires et de clés étrangères.
  - **Remarque :** Les équijointures sont également appelées *jointures simples* ou *jointures internes*.

Extraire des enregistrements à l'aide d'équijointures


```
SELECT employee_id, last_name,
          e.department_id, d.department_id,
          location_id
FROM   employees e, departments d
WHERE  e.department_id = d.department_id;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1800
202	Fay	20	20	1800
124	Mourgos	50	50	1500
141	Rajs	50	50	1500
142	Davies	50	50	1500
143	Matos	50	50	1500
144	Vargas	50	50	1500

...

19 rows selected.

Autres conditions de recherche utilisant l'opérateur AND

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
Whalen	10
Hartstein	20
Fay	20
Mourgos	50
Rajs	50
Davies	50
Matos	50
Vargas	50
Hunold	60
Ernst	60

…

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
20	Marketing
50	Shipping
50	Shipping
50	Shipping
50	Shipping
50	Shipping
60	IT
60	IT

…

Outre la jointure, vous pouvez utiliser des critères dans la clause WHERE pour réduire le nombre de lignes d'une ou de plusieurs tables à prendre en considération. Par exemple, pour afficher le nom et le numéro du service de l'employé Matos, vous devez inclure une condition supplémentaire dans la clause WHERE.

```
SELECT last_name, employees.department_id,department_name
FROM   employees, departments WHERE  employees.department_id =
departments.department_id AND    last_name = 'Matos';
```

Différencier les noms de colonne

- Utilisez des préfixes qui précisent le nom de la table pour différencier les noms de colonne appartenant à plusieurs tables.
- L'utilisation de préfixes désignant la table améliore les performances.
- Différenciez des colonnes de même nom appartenant à plusieurs tables en utilisant des alias de colonne.

Utiliser des alias de table

- Simplifiez les interrogations à l'aide des alias de table.
- L'utilisation de préfixes désignant la table améliore les performances.

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e , departments d
WHERE  e.department_id = d.department_id;
```

Joindre plus de deux tables

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
King	90
Kochhar	90
De Haan	90
Hunold	60
Ernst	60
Lorentz	60
Mourgos	50
Rajs	50
Davies	50
Matos	50
Vargas	50
Zlotkey	80
Abel	80
Taylor	80

...

20 rows selected.

DEPARTMENTS

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

LOCATIONS

LOCATION_ID	CITY
1400	Southlake
1500	South San Francisco
1700	Seattle
1800	Toronto
2500	Oxford

Pour joindre *n* tables entre elles, il faut au minimum n-1 conditions de jointure. Par exemple, deux jointures au moins sont nécessaires pour joindre trois tables.

Joindre plus de deux tables


```
SELECT employee_id, last_name,
          e.department_id, d.department_id,
          l.location_id
FROM   employees e, departments d, location l
WHERE  e.department_id = d.department_id
AND    d.location_id=l.location_id;
```

Les colonnes DEPARTMENT_ID et LOCATION_ID étant communes aux deux tables, vous devez faire précéder leurs nom du nom de la table d'appartenance afin d'éviter toute ambiguïté.

Synthèse

Ce chapitre vous a permis d'apprendre à utiliser des jointures afin d'afficher des données provenant de plusieurs tables en respectant :

Prof. Omar EL BEGGAR

AGRÉGER DES DONNÉES À L'AIDE DE FONCTIONS DE GROUPE

Objectifs

A la fin de ce chapitre, vous pourrez :

- identifier les fonctions de groupe disponibles
- expliquer l'utilisation des fonctions de groupe
- regrouper des données à l'aide de la clause `GROUP BY`
- inclure ou exclure des groupes de lignes à l'aide de la clause `HAVING`

Définition des fonctions de groupe

Les fonctions de groupe agissent sur des groupes de lignes et donnent un résultat par groupe.

EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600
	7000
10	4400

…

20 rows selected.

Salaire maximum dans la table EMPLOYEES.

MAX(SALARY)
24000

Types de fonction de groupe

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE

Syntaxe des fonctions de groupe


```
SELECT          [column,] group function(column), ...
FROM            table
[WHERE          condition]
[GROUP BY       column]
[ORDER BY       column];
```

Utiliser les fonctions AVG et SUM

Les fonctions AVG et SUM s'utilisent avec des données numériques.

```
SELECT  AVG(salary), MAX(salary),
        MIN(salary), SUM(salary)
FROM    employees
WHERE   job_id LIKE '%REP%';
```

AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
8150	11000	6000	32600

L'exemple de la diapositive affiche la moyenne, le maximum, le minimum et la somme des salaires mensuels de tous les représentants.

Utiliser les fonctions MIN et MAX

Les fonctions MIN et MAX s'utilisent avec tous les types de données.

```
SELECT MIN(hire_date), MAX(hire_date)
FROM   employees;
```

MIN(HIRE_	MAX(HIRE_
17-JUN-87	29-JAN-00

Vous pouvez utiliser les fonctions MAX et MIN pour tous les types de données. La diapositive ci-dessus affiche la date d'embauche du dernier employé recruté et celle de l'employé présentant le plus d'ancienneté.

Utiliser la fonction COUNT

La fonction COUNT(*) renvoie le nombre de lignes d'une table

```
SELECT COUNT(*)
FROM   employees
WHERE  department_id = 50;
```

COUNT(*)
5

La fonction COUNT se présente sous trois formes :

COUNT(*)

COUNT(*expr*)

COUNT(DISTINCT *expr*)

La fonction COUNT(*) renvoie le nombre de lignes d'une table, y compris les doublons et les lignes contenant des valeurs NULL.

En revanche, la fonction COUNT(*expr*) renvoie le nombre de valeurs non NULL contenues dans la colonne identifiée par *expr*.

La fonction COUNT(DISTINCT *expr*) renvoie le nombre de valeurs non NULL uniques contenues dans la colonne identifiée par *expr*.

Utiliser la fonction COUNT

- La fonction `COUNT(expr)` renvoie le nombre de lignes contenant des valeurs non NULL dans la colonne *expr*.
- Affichez le nombre de valeurs contenues dans la colonne `DEPARTMENT_ID` de la table `EMPLOYEES`, à l'exception des valeurs NULL.

```
SELECT COUNT(commission_pct)
FROM   employees
WHERE  department_id = 80;
```

COUNT(COMMISSION_PCT)
3

Utiliser le mot-clé DISTINCT

- La fonction COUNT(DISTINCT expr) renvoie le nombre de valeurs non NULL distinctes de la colonne *expr*.
- Affichez le nombre de services distincts contenus dans la table EMPLOYEES.

```
SELECT COUNT(DISTINCT department_id)
FROM   employees;
```

COUNT(DISTINCTDEPARTMENT_ID)
7

Fonctions de groupe et valeurs NULL

Les fonctions de groupe ignorent les valeurs NULL des colonnes.

```
SELECT AVG(commission_pct)
FROM   employees;
```

AVG(COMMISSION_PCT)
.2125

Toutes les fonctions de groupe ignorent les valeurs NULL des colonnes. Dans l'exemple de la diapositive, la moyenne est calculée *uniquement* sur les lignes pour lesquelles la colonne COMMISSION_PCT est correctement renseignée. Le calcul de la moyenne s'effectue par division du total des commissions versées à tous les employés par le nombre d'employés touchant une commission.

Créer des groupes de données

EMPLOYEES

DEPARTMENT_ID	SALARY
10	4400
20	13000
20	6000
50	5800
50	3500
50	3100
50	2500
50	2600
60	9000
60	6000
60	4200
80	10500
80	8600
80	11000
90	24000
90	17000

...

20 rows selected.

4400

9500

3500

6400

10033

Salaire moyen par service dans la table EMPLOYEES.

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10150
	7000

Créer des groupes de données : syntaxe de la clause GROUP BY


```
SELECT          column, group_function(column)
FROM            table
[WHERE          condition]
[GROUP BY       group_by_expression]
[ORDER BY       column];
```

La clause GROUP BY permet d'organiser les lignes d'une table en groupes restreints.

Utiliser la clause GROUP BY

La clause GROUP BY doit inclure toutes les colonnes de la liste SELECT qui ne figurent pas dans des fonctions de groupe.

```
SELECT   department_id, AVG(salary)
FROM     employees
GROUP BY department_id ;
```

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10150
	7000

8 rows selected.

Utiliser la clause GROUP BY

La colonne GROUP BY ne doit pas nécessairement figurer dans la liste SELECT.

```
SELECT   AVG(salary)
FROM     employees
GROUP BY department_id ;
```

AVG(SALARY)
4400
9500
3500
6400
10033.3333
19333.3333
10150
7000

Créer des sous-groupes

EMPLOYEES

DEPARTMENT_ID	JOB_ID	SALARY
90	AD_PRES	24000
90	AD_VP	17000
90	AD_VP	17000
60	IT_PROG	9000
60	IT_PROG	6000
60	IT_PROG	4200
50	ST_MAN	5800
50	ST_CLERK	3500
50	ST_CLERK	3100
50	ST_CLERK	2600
50	ST_CLERK	2500
80	SA_MAN	10500
80	SA_REP	11000
80	SA_REP	8600
...		
20	MK_REP	6000
110	AC_MGR	12000
110	AC_ACCOUNT	8300

20 rows selected.

Dans la table EMPLOYEES, calcul du total des salaires pour chaque poste, au sein de chaque service.

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
50	ST_CLERK	11700
50	ST_MAN	5800
60	IT_PROG	19200
80	SA_MAN	10500
80	SA_REP	19600
90	AD_PRES	24000
90	AD_VP	34000
110	AC_ACCOUNT	8300
110	AC_MGR	12000
	SA_REP	7000

13 rows selected.

Utiliser la clause GROUP BY sur plusieurs colonnes


```
SELECT   department_id dept_id, job_id, SUM(salary)
FROM     employees
GROUP BY department_id, job_id ;
```

DEPT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
50	ST_CLERK	11700
50	ST_MAN	5800
60	IT_PROG	19200
80	SA_MAN	10500
80	SA_REP	19600
90	AD_PRES	24000
90	AD_VP	34000
110	AC_ACCOUNT	8300
110	AC_MGR	12000
	SA_REP	7000

13 rows selected.

Sous-groupes:

Vous pouvez obtenir des informations synthétisées sur des groupes et des sous-groupes en précisant plusieurs colonnes GROUP BY.

Erreurs d'utilisation des fonctions de groupe dans une interrogation

Toute colonne ou expression de la liste SELECT autre qu'une fonction d'agrégation doit être incluse dans la clause GROUP BY.

```
SELECT department_id, COUNT(last_name)
FROM   employees;
```

```
SELECT department_id, COUNT(last_name)
       *
ERROR at line 1:
ORA-00937: not a single-group group function
```

Colonne manquante dans la clause GROUP BY

Erreurs d'utilisation des fonctions de groupe dans une interrogation

- Vous ne pouvez pas utiliser la clause WHERE pour limiter les groupes.
- Utilisez la clause HAVING.
- Vous ne pouvez pas utiliser de fonctions de groupe dans la clause WHERE.

```
SELECT    department_id, AVG(salary)
FROM      employees
WHERE     AVG(salary) > 8000
GROUP BY department_id;
```

```
WHERE  AVG(salary) > 8000
       *
ERROR at line 3:
ORA-00934: group function is not allowed here
```

N'utilisez pas la clause WHERE pour limiter les groupes

Exclure des groupes de résultats

EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600

...

20	6000
110	12000
110	8300

20 rows selected.

Salaire maximum par service, à condition qu'il soit supérieur à 10 000 Dhs

DEPARTMENT_ID	MAX(SALARY)
20	13000
80	11000
90	24000
110	12000

Exclure des groupes de résultats : clause HAVING

Utilisez la clause HAVING pour restreindre les groupes.

1. Les lignes sont regroupées.
2. La fonction de groupe est appliquée.
3. Les groupes qui correspondent à la clause HAVING s'affichent.

```
SELECT       column, group_function
FROM         table
[WHERE       condition]
[GROUP BY    group_by_expression]
[HAVING      group_condition]
[ORDER BY    column];
```

Utiliser la clause HAVING


```
SELECT   department_id, MAX(salary)
FROM     employees
GROUP BY department_id
HAVING   MAX(salary)>10000 ;
```

DEPARTMENT_ID	MAX(SALARY)
20	13000
80	11000
90	24000
110	12000

Utiliser la clause HAVING


```
SELECT    job_id, SUM(salary) PAYROLL
FROM      employees
WHERE     job_id NOT LIKE '%REP%'
GROUP BY job_id
HAVING    SUM(salary) > 13000
ORDER BY SUM(salary);
```

JOB_ID	PAYROLL
IT_PROG	19200
AD_PRES	24000
AD_VP	34000

Imbriquer des fonctions de groupe

Affichez le salaire moyen maximum.

```
SELECT   MAX(AVG(salary))
FROM     employees
GROUP BY department_id;
```

MAX(AVG(SALARY))
19333.3333

Synthèse

Ce chapitre vous à permis d'apprendre à :

- utiliser les fonctions de groupe COUNT, MAX, MIN, AVG
- écrire des instructions contenant la clause GROUP BY
- écrire des intructions contenant la clause HAVING

```
SELECT        column, group_function(column)
FROM          table
[WHERE        condition]
[GROUP BY     group_by_expression]
[HAVING       group_condition]
[ORDER BY     column];
```

Prof. Omar EL BEGGAR

MANIPULER DES DONNÉES

Objectifs

A la fin de ce chapitre, vous pourrez :

- décrire chaque instruction LMD
- insérer des lignes dans une table
- modifier des lignes dans une table
- supprimer des lignes d'une table

Langage de manipulation de données

- Une instruction LMD est exécutée lorsque vous :
  - ajoutez des lignes à une table,
  - modifiez des lignes existantes dans une table,
  - supprimez des lignes d'une table.

Ajouter une nouvelle ligne dans une table

70	Public Relations	100	1700

Nouvelle ligne

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

… insérer une nouvelle ligne dans la table DEPARMENTS …



DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700
70	Public Relations	100	1700

Syntaxe de l'instruction INSERT

- L'instruction INSERT permet d'ajouter de nouvelles lignes dans une table.

```
INSERT INTO   table [(column [, column...])]
VALUES        (value [, value...]);
```

- Cette syntaxe n'insère qu'une seule ligne à la fois.

Insérer de nouvelles lignes

- Insérez une nouvelle ligne en précisant une valeur pour chaque colonne.
- Indiquez les valeurs dans l'ordre par défaut des colonnes dans la table.
- Indiquez éventuellement les colonnes dans la clause INSERT.

```
INSERT INTO departments(department_id, department_name,
                        manager_id, location_id)
VALUES      (70, 'Public Relations', 100, 1700);
1 row created.
```

- Placez les valeurs de type caractère et date entre apostrophes.

Insérer des lignes contenant des valeurs NULL

- Méthode implicite : n'indiquez pas la colonne dans la liste.

```
INSERT INTO   departments (department_id,
                           department_name  )
VALUES        (30, 'Purchasing');
1 row created.
```

- **Méthode explicite : indiquez le mot-clé `NULL` dans la clause `VALUES`.**

```
INSERT INTO   departments
VALUES        (100, 'Finance', NULL, NULL);
1 row created.
```

Insérer des valeurs spéciales

La fonction SYSDATE enregistre la date et l'heure en cours.

```
INSERT INTO employees (employee_id,
                 first_name, last_name,
                 email, phone_number,
                 hire_date, job_id, salary,
                 commission_pct, manager_id,
                 department_id)
VALUES          (113,
                 'Louis', 'Popp',
                 'LPOPP', '515.124.4567',
                 SYSDATE, 'AC_ACCOUNT', 6900,
                 NULL, 205, 100);
1 row created.
```

Modifier les données d'une table

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSION_F
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	60	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	60	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	60	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

Mettez à jour les lignes de la table EMPLOYEES.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSIO
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	30	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	30	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	30	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

Syntaxe de l'instruction UPDATE

- Utilisez l'instruction UPDATE pour modifier des lignes existantes.

```
UPDATE        table
SET           column = value [, column = value, ...]
[WHERE        condition];
```

- Si nécessaire, vous pouvez modifier plusieurs lignes à la fois.

Modifier des lignes d'une table

- La clause WHERE permet de modifier une ou plusieurs lignes spécifiques.

```
UPDATE employees
SET    department_id = 70
WHERE  employee_id = 113;
1 row updated.
```

- En cas d'absence de la clause WHERE, toutes les lignes sont modifiées.

```
UPDATE   copy_emp
SET      department_id = 110;
22 rows updated.
```

Modifier deux colonnes à l'aide d'une sous-interrogation

Modifiez le poste et le salaire de l'employé 114 pour qu'ils correspondent à ceux de l'employé 205.

```
UPDATE   employees
SET      job_id  = (SELECT  job_id
                    FROM    employees
                    WHERE   employee_id = 205),
         salary  = (SELECT  salary
                    FROM    employees
                    WHERE   employee_id = 205)
WHERE    employee_id    =  114;
1 row updated.
```

Modifier des lignes en fonction d'une autre table

Utilisez des sous-interrogations dans l'instruction UPDATE pour modifier des lignes d'une table à l'aide de valeurs d'une autre table.

```
UPDATE   copy_emp
SET      department_id  =   (SELECT department_id
                             FROM employees
                             WHERE employee_id = 100)
WHERE    job_id         =   (SELECT job_id
                             FROM employees
                             WHERE employee_id = 200);
1 row updated.
```

Erreur de contrainte d'intégrité lors de la modification de lignes


```
UPDATE employees
SET    department_id = 55
WHERE  department_id = 110;
```

```
UPDATE employees
       *
ERROR at line 1:
ORA-02291: integrity constraint (HR.EMP_DEPT_FK)
violated - parent key not found
```

Le numéro de service 55 n'existe pas.

Supprimer une ligne d'une table

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
100	Finance		
50	Shipping	124	1500
60	IT	103	1400

Supprimez une ligne de la table DEPARTMENTS.

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
50	Shipping	124	1500
60	IT	103	1400

Instruction DELETE

Vous pouvez supprimer des lignes d'une table au moyen de l'instruction DELETE.

```
DELETE [FROM]   table
[WHERE          condition];
```

Supprimer des lignes d'une table

- La clause WHERE permet de supprimer des lignes spécifiques.

```
 DELETE FROM departments
 WHERE  department_name = 'Finance';
1 row deleted.
```

- En cas d'absence de la clause WHERE, toutes les lignes sont supprimées.

```
DELETE FROM  copy_emp;
22 rows deleted.
```

Supprimer des lignes associées à des valeurs d'une autre table

Utilisez des sous-interrogations dans l'instruction DELETE pour supprimer des lignes dont certaines valeurs correspondent à celles d'une autre table.

```
DELETE FROM employees
WHERE  department_id =
                        (SELECT department_id
                         FROM   departments
                         WHERE  department_name LIKE '%Public%');
1 row deleted.
```

Erreur de contrainte d'intégrité lors de la suppression de lignes


```
DELETE FROM departments
WHERE       department_id = 60;
```

```
DELETE FROM departments
            *
ERROR at line 1:
ORA-02292: integrity constraint (HR.EMP_DEPT_FK)
violated - child record found
```

Vous ne pouvez pas supprimer une ligne qui contient une clé primaire utilisée comme clé étrangère dans une autre table.

Synthèse

Ce chapitre vous a permis d'apprendre à utiliser des instructions LMD.

Instruction	Description
INSERT	Ajoute une nouvelle ligne dans une table
UPDATE	Modifie des lignes dans une table
DELETE	Supprime des lignes d'une table

SOUS-INTERROGATIONS

Objectifs

A la fin de ce chapitre, vous pourrez :

- décrire les types de problème que les sous-interrogations permettent de résoudre
- définir des sous-interrogations
- énumérer les types de sous-interrogation
- écrire des interrogations monolignes et multilignes

Résoudre un problème à l'aide d'une sous-interrogation

Qui touche un salaire supérieur à celui d'Abel ?

Interrogation principale :



Quels employés touchent un salaire supérieur à celui d'Abel ?

Sous-interrogation :



Quel est le salaire d'Abel ?

Syntaxe des sous-interrogations


```
SELECT    select_list
FROM      table
WHERE     expr operator
                    (SELECT        select_list
                     FROM          table);
```

- La sous-interrogation (interrogation interne) s'exécute une fois avant l'interrogation principale.
- L'interrogation principale (interrogation externe) utilise le résultat de la sous-interrogation.

Utiliser une sous-interrogation


```
SELECT last_name
FROM   employees
WHERE  salary >
                (SELECT salary
                 FROM   employees
                 WHERE  last_name = 'Abel');
```

11000

LAST_NAME
King
Kochhar
De Haan
Hartstein
Higgins

Règles d'utilisation des sous-interrogations

- Placez les sous-interrogations entre parenthèses.
- Placez les sous-interrogations dans la partie droite de la condition de comparaison.
- La clause `ORDER BY` de la sous-interrogation n'est requise que si vous effectuez une analyse de type n-premiers.
- Utilisez des opérateurs monolignes dans les sous-interrogations monolignes et des opérateurs multilignes dans les sous-interrogations multilignes.

Types de sous-interrogation

- **Sous-interrogation monoligne**



- **Sous-interrogation multiligne**



Sous-interrogations monolignes

- Renvoient une seule ligne
- Utilisent des opérateurs de comparaison monolignes

Opérateur	Signification
=	Egal à
>	Supérieur à
>=	Supérieur ou égal à
<	Inférieur à
<=	Inférieur ou égal à
<>	Différent de

Exécuter des sous-interrogations monolignes


```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id =
                (SELECT job_id
                 FROM   employees
                 WHERE  employee_id = 141)
AND    salary >
                (SELECT salary
                 FROM   employees
                 WHERE  employee_id = 143);
```


LAST_NAME	JOB_ID	SALARY
Rajs	ST_CLERK	3500
Davies	ST_CLERK	3100

Utiliser des fonctions de groupe dans une sous-interrogation


```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary =
                (SELECT MIN(salary)
                 FROM   employees);
```

2500

LAST_NAME	JOB_ID	SALARY
Vargas	ST_CLERK	2500

Clause HAVING et sous-interrogations

- Le serveur Oracle exécute d'abord les sous-interrogations.
- Le serveur Oracle renvoie les résultats dans la clause HAVING de l'interrogation principale.

```
SELECT   department_id, MIN(salary)
FROM     employees
GROUP BY department_id
HAVING   MIN(salary) >
                        (SELECT MIN(salary)
                         FROM   employees
                         WHERE  department_id = 50);
```

2500

Erreurs dans les sous-interrogations


```
SELECT employee_id, last_name
FROM   employees
WHERE  salary =
                (SELECT   MIN(salary)
                 FROM     employees
                 GROUP BY department_id);
```

```
ERROR : single-row subquery returns more than
one row
```

Opérateur monoligne dans une sous-interrogation multiligne

Problèmes liés aux sous-interrogations


```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
                (SELECT job_id
                 FROM   employees
                 WHERE  last_name = 'Haas');
```

```
no rows selected
```

La sous-interrogation ne renvoie aucune valeur

Sous-interrogations multilignes

- Renvoient plusieurs lignes
- Utilisent des opérateurs de comparaison multilignes

Opérateur	Signification
IN	Egal à n'importe quel membre de la liste
ANY	Compare la valeur à chaque valeur renvoyée par la sous-interrogation
ALL	Compare la valeur à toutes les valeurs renvoyées par la sous-interrogation

Utiliser l'opérateur ANY dans les sous-interrogations multilignes


```
SELECT employee_id, last_name, job_id, salary
FROM   employees        9000, 6000, 4200
WHERE  salary < ANY
                    (SELECT salary
                     FROM   employees
                     WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
124	Mourgos	ST_MAN	5800
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

...

10 rows selected.

Utiliser l'opérateur ALL dans les sous-interrogations multilignes


```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ALL
                    (SELECT salary
                     FROM   employees
                     WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

9000, 6000, 4200

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

Synthèse

Ce chapitre vous à permis d'apprendre à :

- déterminer quand une sous-interrogation peut aider à résoudre un problème
- écrire des sous-interrogations lorsqu'une interrogation est basée sur des valeurs inconnues

```
SELECT   select_list
FROM     table
WHERE    expr operator
                   (SELECT select_list
                    FROM   table);
```

UNIVERSITÉ HASSAN II -CASABLANCA

Faculté des Sciences et Techniques Mohammedia



Module Informatique I 231:

Bases de Données

Parcours BCG

2014-2015

Prof. Omar EL BEGGAR

CHAPITRE 9:

CONCEPTION D'UNE BASE DE DONNEES

Sommaire

- Pourquoi Conception d'une BDR
- Phases de conception d'une BDR
- Méthodologie de Conception
- Modèle Conceptuel de Données (MCD)
- Aperçu sur la méthode Merise
- Concept Entité
- Concept Attribut et Identifiant
- Concept Association
- Normalisation du MCD
- Conclusion

Pourquoi Conception d'une BDR

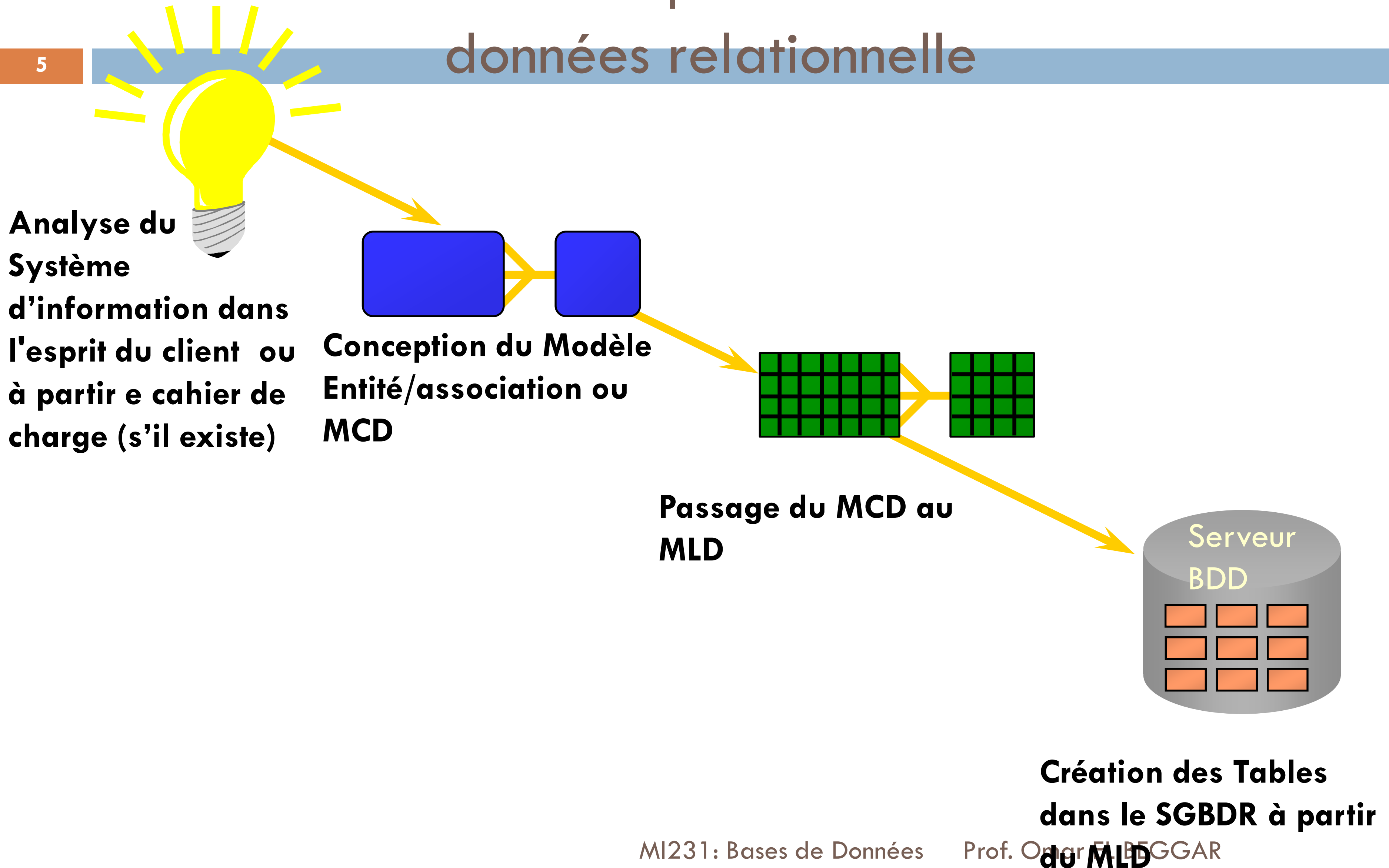
- Quand nous construisons directement les tables d'une base de données dans un logiciel de gestion de bases de données(Oracle, MySQL, DB2, PostGre, Informix,…), nous sommes exposés à deux types de problèmes:

- Nous ne savons pas toujours dans quelle table placer certaines colonnes(par exemple l'adresse de livraison se met dans la table des clients ou dans la table des commandes?);

- Nous avons du mal à prévoir les tables de jonction intermédiaires(par exemple, la table des interprétations qui est indispensable entre les tables des films et la table des acteurs).

Il est donc nécessaire de recourir à une étape préliminaire de conception.

Phases de conception d’une Base de données relationnelle



Phases de conception d'une base de données

Quatre phases :

1. Analyse du problème
2. Modèle conceptuel des données (MCD)
3. Modèle logique des données (MLD)
4. Modèle physique (réalisation dans le SGBD)

Méthodologie de Conception

- Face à une situation bien définie(soit à travers un énoncé précis, soit à travers une collection de formulaires ou d'états ou à travers un cahier de charge,…), vous pouvez suivre cette méthodologie afin de concevoir votre base de données et établir le modèle Entité/Association:

1. Identifier les entités (faire attention aux synonymes),
2. Lister leurs attributs,
3. Choisir un identifiant ou créer un,
4. Établir les associations et ajouter leurs attributs s'ils existent,
5. Calculer les cardinalités,
6. Normaliser votre modèle.

Modèle Conceptuel de Données

- Modèle Conceptuel de Données (MCD) est une représentation graphique formalisée avec des entités et associations, qui tente de représenter et historier les données métier d'un système d'information.
- Ce schéma obéit à quelques conventions graphiques très simples et à quelques règles de construction ou de normalisation précises.
- Il manipule essentiellement deux concepts : les **entités et les associations**
- Il fait partie des modèles conceptuels proposés par la méthode d’analyse et conception des systèmes d’information française **« Merise ».**

Aperçu sur la méthode Merise

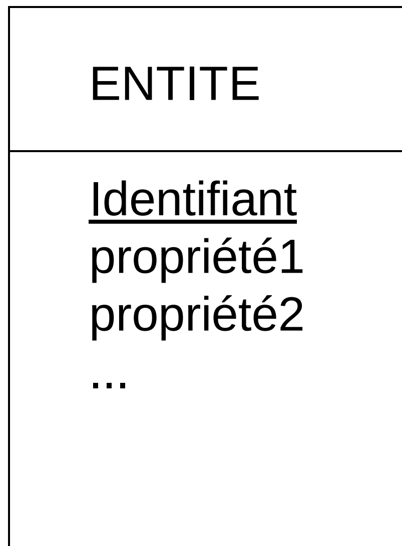
- Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise (Merise), élaborée en France en 1979, pour le compte du ministre de l'industrie français.
- Merise est une méthode d'analyse et de conception des Systèmes d'Information (SI) des entreprises
- La démarche Merise étudie le SI d'une entreprise en procédant à 3 découpages sur 4 niveaux: MERISE décrit le SI sous forme de trois découpages : communication, traitement et données et Quatre niveaux : conceptuel, organisationnel, logique et physique

Entité-Définition-

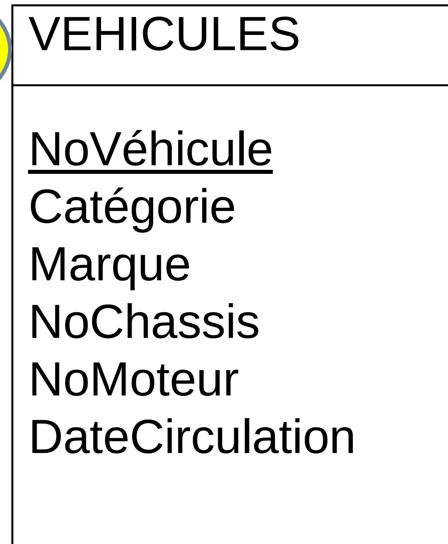
- Une entité forment un tout qui regroupe des **occurrences de même nature.**
- **Toutes** les occurrences d'une entité sont décrites par un ensemble de **propriétés dont les valeurs** changent d'une occurrence à l'autre.
- Elles représentent soit une personne physique (les professeurs, les étudiants), soit une personne morale (les entreprises), soit une chose (les compétences, les types de stage, les promos), soit des événements (les stages).

Entité-Représentation-

- Elle est représentée tout simplement par un **rectangle** composé de deux compartiments le premier compartiment indique le nom de l'entité et le deuxième compartiment contient la liste de toutes ses propriétés ou attributs.

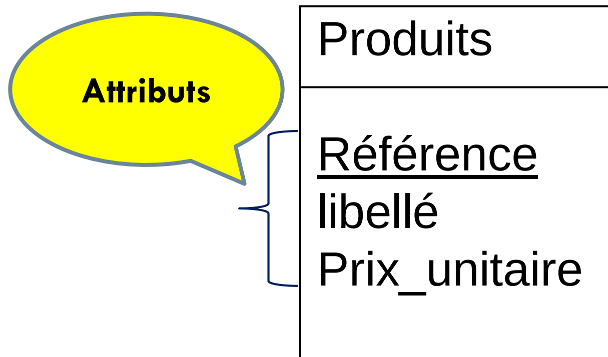


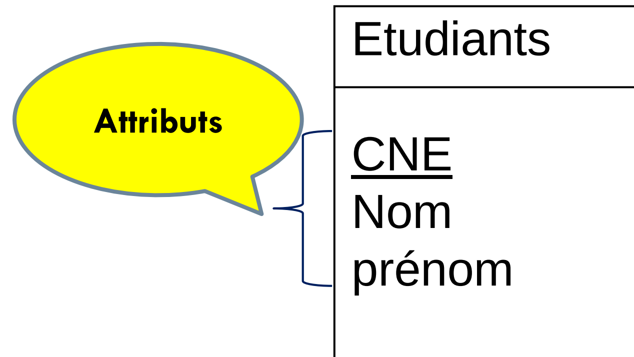
Exemple



Attribut

- Un attribut est une propriété d'une entité ou d'une association.
- Par exemple: Référence, libellé et prix unitaire se sont des attributs de l'entité produit. Alors que le CNE, nom et prénom se sont des attributs de l'entité Etudiant.
- Un attribut a un nom et un type de données dont ses valeurs doivent le respecter.
- L'entité et ses attributs doivent traiter qu'un seul sujet afin d'assurer une certaine cohérence du modèle.





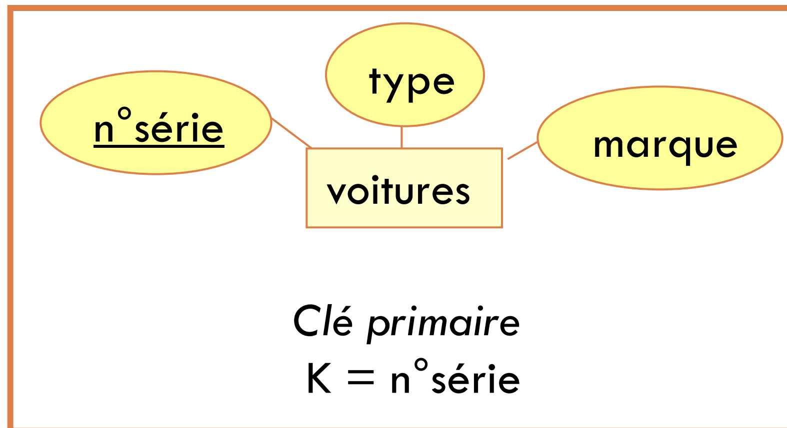
Identifiant ou clé

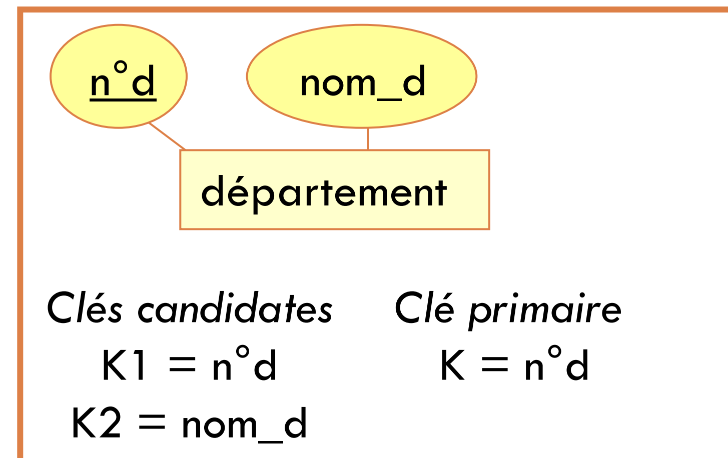
- Parmi les propriétés une (ou une combinaison de plusieurs) joue un rôle particulier car elle permet d'identifier à coup et de façon unique une occurrence de l'entité: c'est l'identifiant dit aussi clé.
- Le plus souvent c'est un numéro, un code, une référence etc. Soit il existe déjà dans la réalité du SI ou le fruit d'une codification interne.
- Exemple le CNE de l'étudiant ou la référence de l'article.
- Toute entité doit avoir un identifiant, en principe celui-ci est stable, c'est à dire que sa valeur pour une occurrence donnée ne change pas.
- Par construction il apparaît en tête des propriétés et il est souligné.



Type d'identifiants ou Clés

- clé candidate: un ensemble minimal d'attributs qui identifie de façon unique une occurrence d'entité
- clé primaire: une clé candidate choisie pour identifier de façon unique chaque occurrence d'entité
- clé composée: une clé candidate composée de deux ou plusieurs attributs





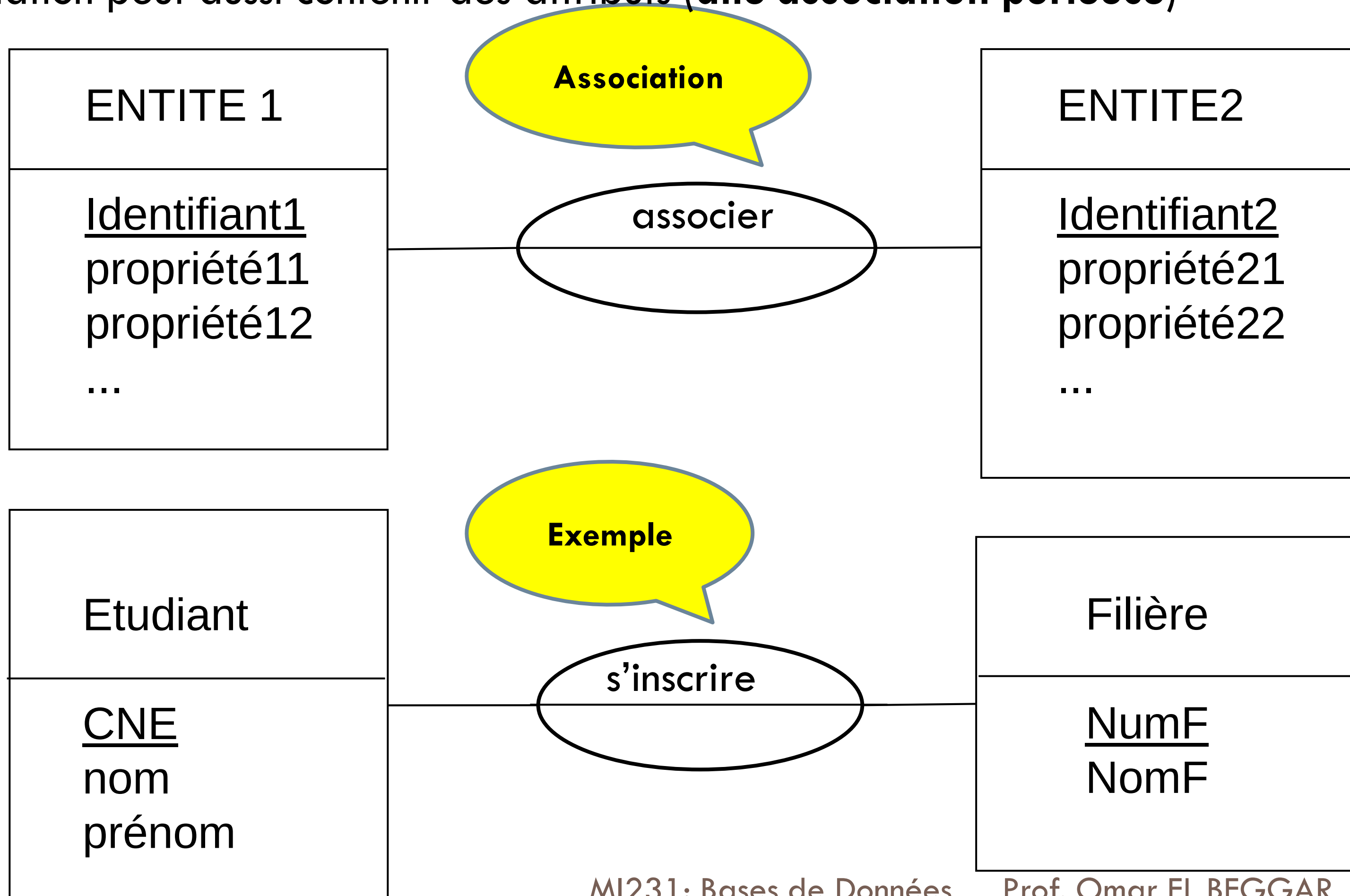
Association-Définition-

- Association est une liaison entre entités qui a une signification précise.
- Il est judicieux de nommer les associations par un verbe à l'infinitif car il y a toujours plusieurs sens de lecture.
- **La plupart des associations sont binaires, c'est à dire qu'elles relient deux entités.**
- Par exemple Effectuer associe étudiant et stage : un stage est effectué par un étudiant et ce dernier peut effectuer plusieurs stages : les deux sens de lecture sont chacun porteur de sens.

Association-Représentation-

Une association est représentée par un ellipse qui porte son nom (souvent un verbe à l'infinitif)

Une association peut aussi contenir des attributs (**dite association porteuse**)

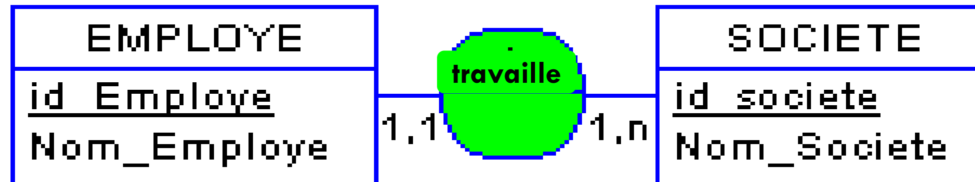


Types d’association

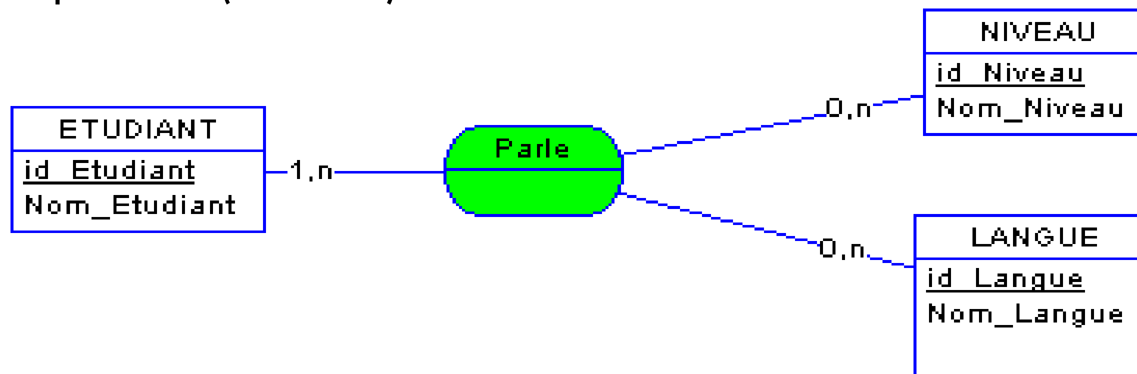
- Association binaire: liaison entre deux entités seulement.
- Association plurielles: Elle relie plusieurs entités à la fois, (une association peut être ternaire, voire quaternaire, au delà c'est beaucoup plus rare…)
- Association réflexive: lie l’entité à elle même

Exemples des types d'associations

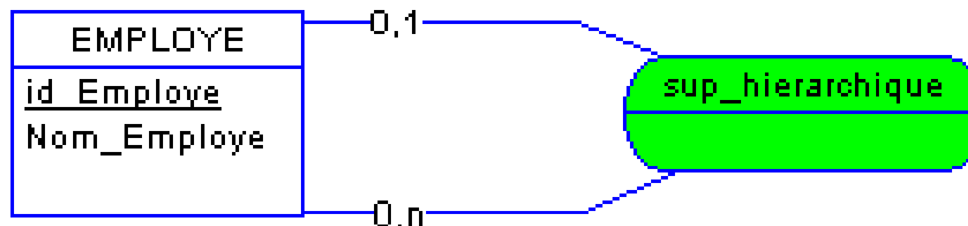
Association binaire



Association plurielle (ternaire)



Association réflexive

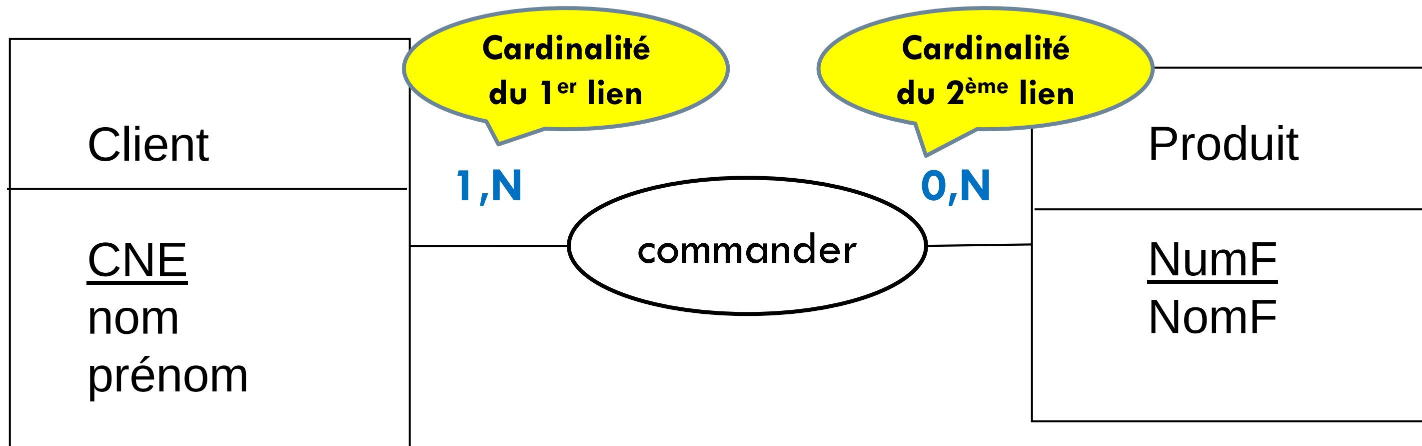


Remarques

- Une entité possède au moins une propriété(qui est son identifiant);
- Une association peut être dépourvue d'attributs

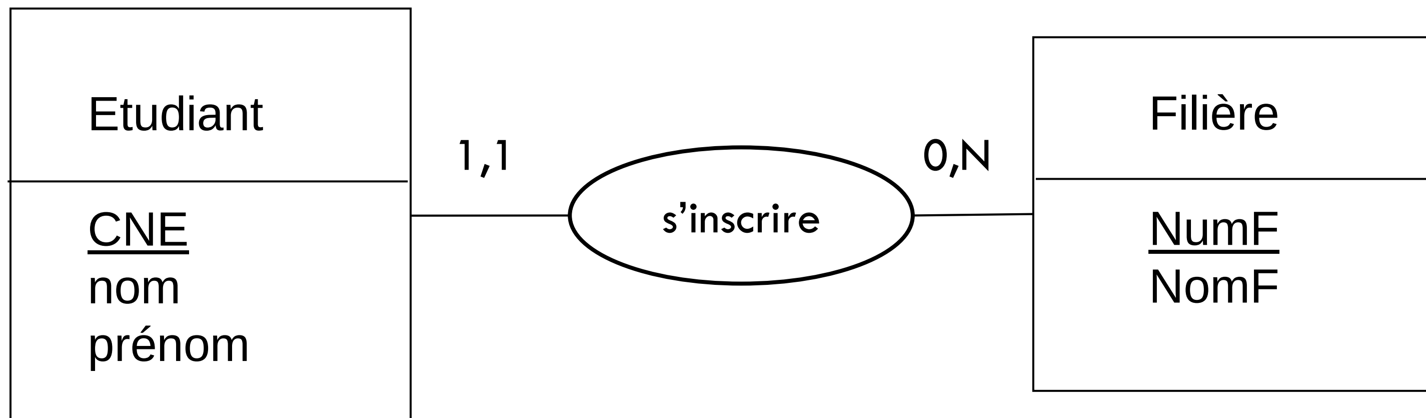
Cardinalité

- La cardinalité d'un lien entre une entité et une association précise le minimum et le maximum de fois qu'une occurrence de l'entité peut être concerné par l'association.
- Exemple : un client a au moins commandé un produit et peut commander n produits (n étant indéterminé), tandis qu'un produit peut avoir été commandé entre 0 et N fois (même si ce n'est pas le même N que précédemment).



Déterminer les cardinalités

- La seule difficulté pour déterminer les cardinalités est de poser les questions dans le bon sens. Autour de l'association effectuer par exemple:
- Côté étudiants, la question est : « un étudiant peut être inscrit à combien de filière? »
- Côté filières, la question est: « une filière peut avoir l'inscription de combien d'étudiants ? »



Normalisation

- Le but essentiel de la normalisation est d'éviter les anomalies transactionnelles pouvant découler d'une mauvaise modélisation des données et ainsi éviter un certain nombre de problèmes potentiels tels que :
  - les anomalies de lecture,
  - les anomalies d'écriture,
  - la redondance des données
  - et la contre performance.
- La normalisation des modèles de données permet de vérifier la robustesse de leur conception pour améliorer la modélisation (et donc obtenir une meilleure représentation) et faciliter la mémorisation des données en évitant les problèmes sous-jacents de mise à jour ou de cohérence.
- La normalisation s'applique à toutes les entités et aux relations porteuses de propriétés.

Normalisation des noms

- Le nom d'une entité, association ou propriété doit être unique.
- Utiliser un nom commun au pluriel pour les entités
- Pour les associations, utiliser un verbe à l'infinitif
- Pour les propriétés, utiliser un nom commun singulier

Normalisation des identifiants

- Eviter les identifiants qui ne garantissent pas l'unicité des occurrences de l'entités
- Eviter les identifiants susceptibles de changer au cours du temps(plaque d'immatriculation)
- Eviter les identifiants longs formée par des chaines de caractères.
- En conclusion l'identifiant ou la clé d'une entité doit être choisie de manière qu'il soit entier simple et inchangeable dans le temps

Normalisation des propriétés

- Eviter les propriétés calculables ou dérivables
- Remplacer les propriétés en plusieurs exemplaires ou admettant plusieurs valeurs par une nouvelle entité contenant cette propriété et associée à l'ancienne entité.

Formes normales

- A ces règles de normalisation, il convient d'ajouter le 3 formes normales

1FN - première forme normale

- Relation dont tous les attributs :
  - contiennent des valeurs indivisibles et non répétitives (les valeurs ne peuvent pas être divisées en plusieurs sous-valeurs dépendant également individuellement de la clé primaire)
  - *Le non respect de deux premières conditions de la 1FN rend la recherche parmi les données plus lente et conduit régulièrement à une mise à jour des données.*
- Dans ce cas les valeurs du fournisseur sont multivaluées et ne sont pas atomiques. Pour que cette relation soit en première forme normale, il faut décomposer les attributs de la colonne fournisseur comme suit :

Produit	Fournisseur
téléviseur	VIDEO SA, HITEK LTD



Produit	Fournisseur
téléviseur	VIDEO SA
téléviseur	HITEK LTD

Exemple de 1FN

Dans ce cas, un employé peut posséder plus qu'un numéro de téléphone. Les anomalies constatées dans ce modèle sont relative:

- À la Recherche lente par exemple d'un numéro de tél.
- La mise à jour complexe d'un numéro de tél.

Employés

N°Emp	nom	NumTél
1	Dupont	{0123456, 061111}
2	Durant	{0234567, 062222}
3	Villier	{0345678, 063333}
12	Fornier	{0456789, 064444}

Correction ?

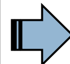
2FN - deuxième forme normale

- la relation respectant la première forme normale et dont :
  - Tout attribut ne composant pas un identifiant dépend de tout l’ identifiant.
  - *Le non respect de la 2FN entraine une redondance des données qui encombrent alors inutilement la mémoire et l'espace disque.*
  - chaque attribut qui n'appartient pas à la clé (l'ensemble des attributs permettant d'identifier de manière unique un tuple de l'entité) ne dépend pas uniquement d'une partie de la clé
- Autrement dit, toute dépendance Clé → A est élémentaire (si A n’appartient pas á une clé).

2FN - deuxième forme normale

- Admettons que la clé de cette table soit une clé composite (produit - fournisseur). Dans le cas d'un changement d'adresse d'un fournisseur, il faudra faire preuve de beaucoup d'attention pour n'oublier aucun endroit où l'adresse est mentionnée. En effet, on constate que le champ adresse ne dépend que d'une partie de la clé : le champ fournisseur, ce qui induit la possibilité d'une redondance au sein de la table. Il convient donc de scinder la table en deux:

Produit	Fournisseur	Adresse fournisseur
téléviseur	VIDEO SA	13 rue du cherche-midi
écran plat	VIDEO SA	13 rue du cherche-midi
téléviseur	HITEK LTD	25 Bond Street



Produit	Fournisseur
téléviseur	VIDEO SA
téléviseur	HITEK LTD
écran plat	VIDEO SA

Fournisseur	Adresse fournisseur
VIDEO SA	13 rue du cherche-midi
HITEK LTD	25 Bond Street

- De cette manière, un changement d'adresse ne donne lieu qu'à une seule modification dans la table des fournisseurs

3FN - troisième forme normale

- la relation respectant la seconde forme normale et dont :
  - Tout attribut ne composant pas un identifiant dépend **directement** d'un identifiant.
  - *Le non respect de la 3FN peut également entrainer une redondance des données.*
  - les attributs qui ne font pas partie de la clé ne dépendent pas d'attributs ne faisant pas non plus partie de la clé (les attributs sont donc complètement indépendants les uns des autres).

Fournisseur	Adresse fournisseur	Ville	Pays
VIDEO SA	13 rue du cherche-midi	PARIS	FRANCE
HITEK LTD	25 Bond Street	LONDON	ENGLAND

  - Le pays de l'adresse n'est pas dépendant de la clé de la table, à savoir le nom du fournisseur, mais est fonction de la ville de l'adresse. De nouveau, il est préférable de scinder la table en deux:

Fournisseur	Adresse fournisseur	Ville
VIDEO SA	13 rue du cherche-midi	PARIS
HITEK LTD	25 Bond Street	LONDON

Ville	Pays
PARIS	FRANCE
LONDON	ENGLAND

Les différentes formes normales

- Pour se souvenir de l'ordre et des caractéristiques des trois premières formes normales, il suffit de se rappeler le serment que tous les témoins doivent prêter devant la justice :

Je jure de dire la vérité, toute la vérité, rien d'autre que la vérité.

Ce qui donne : 1FN = La clé. 2FN = Toute la clé. 3FN = Rien que la clé. La phrase originale étant : *"The key, the whole key, nothing but the key"* (Chris Date). Elle est empruntée à l'œuvre de Shakespeare.

Bibliographie

- Philippe Rigaux "Pratique de MySQL et PHP", 2nde édition, O'Reilly, 2003
- Cyril GRUAU, 13 Juillet 2006, Conception d'une base de données
- Gabriella Salzano - UMLV , Mars 2007



UNIVERSITÉ HASSAN II- CASABLANCA
Faculté des Sciences et
Techniques Mohammedia

Module Informatique I 231:

Bases de Données

Parcours BCG

2014-2015

Prof. Omar EL BEGGAR

CHAPITRE 10:

PASSAGE DU MODELE ENTITE/ASSOCIATION AU MODELE RELATIONNEL

INTRODUCTION AU MLD

- **Après avoir conçu le Modèle Conceptuel de Donnée (MCD), il est maintenant temps de le transposer en Modèle Logique de Données Relationnelles (MLDR). Ce MLDR est en fait le dernier pas avant la création de la base de données.**
- **Après avoir définis les notions de clé primaire et de clé étrangère, nous étudierons plus particulièrement les 6 règles strictes, nécessaires et suffisantes pour passer d'un MCD à un MLD.**
- **Le MLD est lui aussi indépendant du matériel et du logiciel, il ne fait que prendre en compte l'organisation des données.**

Représentation d'un Modèle relationnel

Le modèle logique de données peut être exprimé sous deux formes :

- Forme textuelle (schéma relationnel)
- Forme graphique

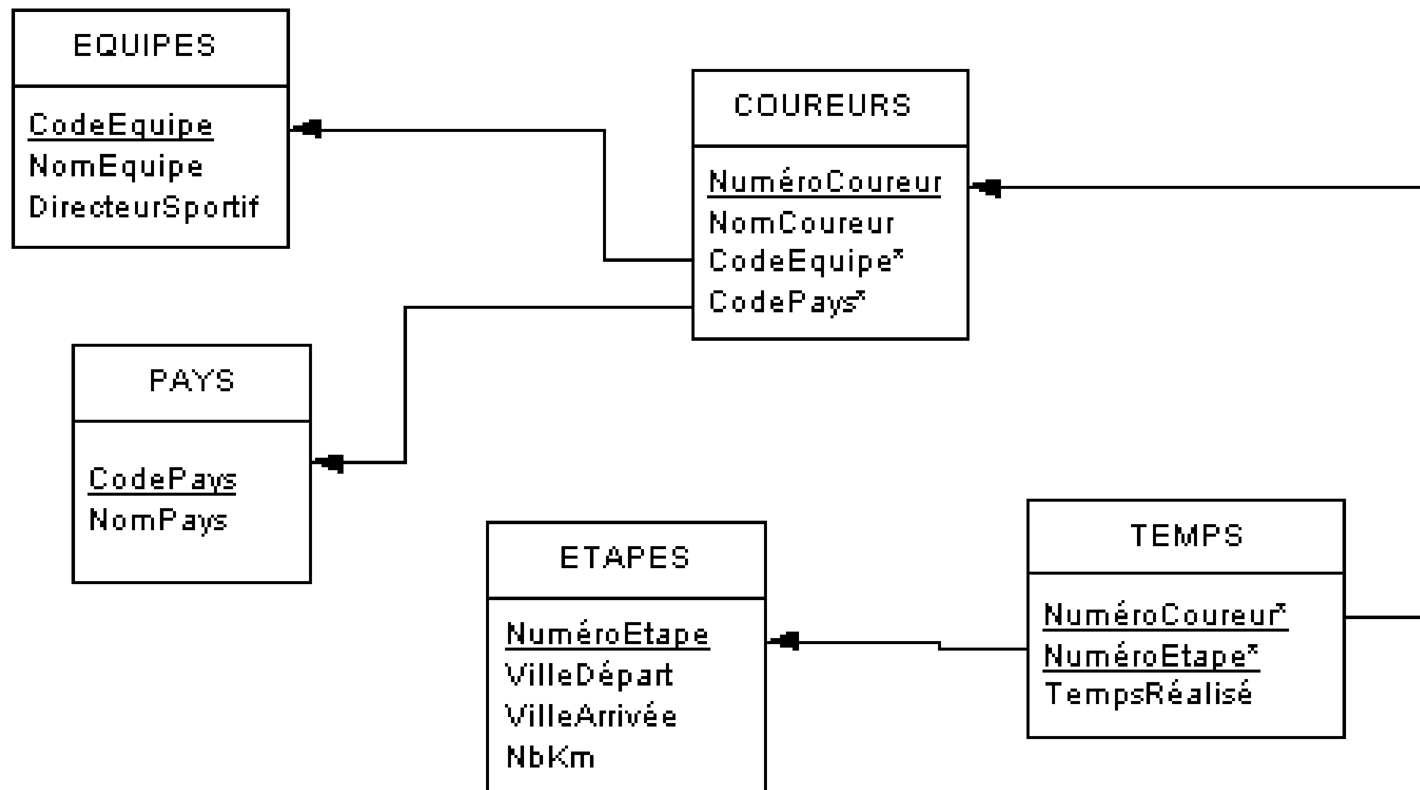
Schéma relationnel

- **le MLDR ou schéma relationnel a été inventé par Codd en 1970, et repose sur la Théorie Ensembliste… vocabulaire : Les données sont stockées dans des relations. Une relation est un ensemble de T-uple, et un T-uple est définis par un ou plusieurs attributs. Dans la pratique, la relation est en fait la table, un T-uple est une ligne (ou enregistrement), et les attributs sont les colonnes.**

- **Schéma relationnel permet de présenter les différentes relations de la BD, ainsi que leurs clés primaires, clés étrangers et les autres attributs.**
- **Exemple :**
- **TABLE(<u>cle_primaire</u>, colonne1, colonne2, #cle_etrangere)**
- **Exemple :**
- Dans la forme textuelle , les clés primaires étant soulignées et les clés
- étrangères marquées par un signe distinctif (ici * ou #) à la fin.
  - EQUIPES (<u>CodeEquipe</u>, NomEquipe, DirecteurSportif)
  - COUREURS (<u>NuméroCoureur</u>, NomCoureur, CodeEquipe*, CodePays*)
  - ETAPES (<u>NuméroEtape</u>, VilleDépart, VilleArrivée, NbKm)
  - TEMPS (<u>NuméroCoureur*, NuméroEtape*</u>, TempsRéalisé)
  - PAYS (<u>CodePays</u>, NomPays)

Forme graphique

- On peut aussi le représenter sous forme graphique, de manière à mieux visualiser et interpréter les liens :



CLES PRIMAIRE ETRANGERE

- **<u>Exemple de la table EMAILS :</u>**

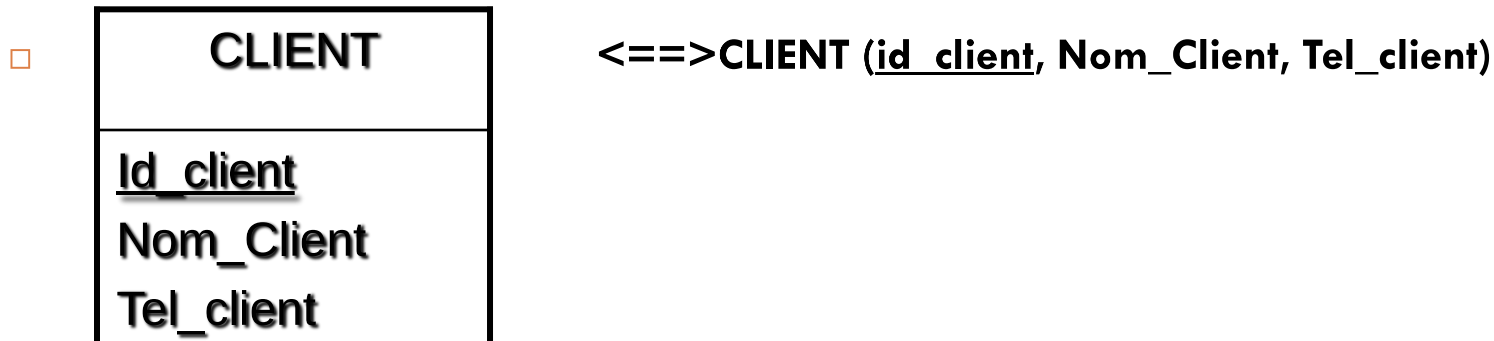
Id_email	Sujet	DateEnvoie	Contenu	Id_rubrique
12	Email12	11/11/2007	Texte	1
13	Email13	01/01/2008	Texte	2

- **Cette table est décrite par :**
  EMAIL (<u>id_email</u>, Sujet, DateEnvoie, Contenu, #id_rubrique)
- **Chaque enregistrement doit être identifié de manière unique. L'attribut qui permet d'identifier de façon unique chaque ligne est appelée la Clé Primaire. Elle peut être composée, c'est à dire comprendre plusieurs attributs. Ici, il s'agit de l'attribut id_email.**
- **La table EMAILS comprend un attribut provenant de la table RUBRIQUES, l'attribut id_rubrique. Cet attribut est appelé Clé Etrangère.**
  Dans le formalisme, la clé primaire est soulignée, et la clé étrangère est précédée du signe #.
  <u>Dans notre exemple :</u>
  Rubrique (<u>id_rubrique</u>, Nom)
  EMAILS (<u>id_email</u>, Sujet, DateEnvoie, Contenu, #id_rubrique)

REGLES DE PASSAGE

- **1 : Une entité se transforme en une relation (table)**

Toute entité du MCD devient une relation du MLDR, et donc une table de la Base de Donnée. Chaque propriété de l'entité devient un attribut de cette relation, et dont une colonne de la table correspondante. L'identifiant de l'entité devient la Clé Primaire de la relation (elle est donc soulignée), et donc la Clé Primaire de la table correspondante.



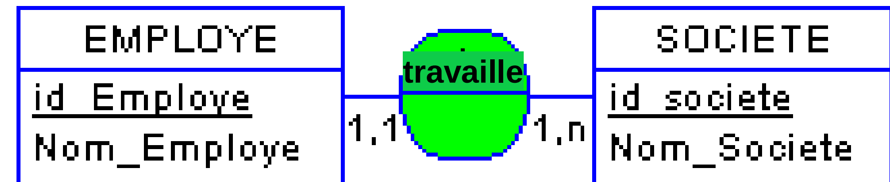
REGLES DE PASSAGE

- **2 : Relation binaire aux cardinalités (X,1) - (X,n), X=0 ou X=1**

  La Clé Primaire de la table à la cardinalité (X,n) devient une Clé Etrangère dans la table à la cardinalité (X,1) :

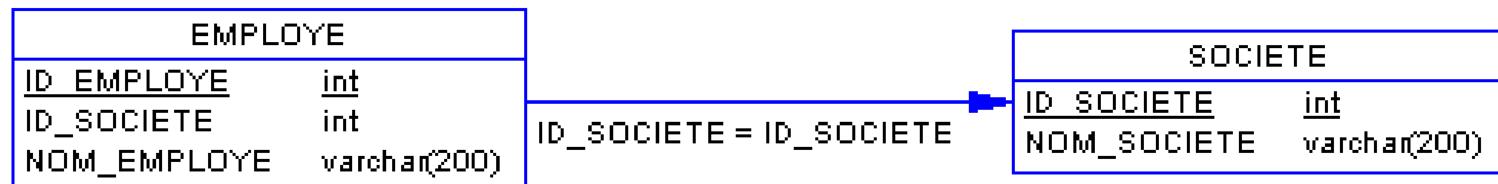
- **<u>Exemple de Système d'Information (SI) :</u>**
  Un employé a une et une seule société. Une société a 1 ou n employés.

- **<u>Modèle Conceptuel de Donnée (MCD) :</u>**



- **<u>Modèle Logique de Donnée Relationnelle (MLDR) :</u>**
  EMPLOYE (<u>id_Employe</u>, Nom_Employe, #id_Societe)
  SOCIETE (<u>id_Societe</u>, Nom_Societe)

- **<u>Représentation graphique :</u>**

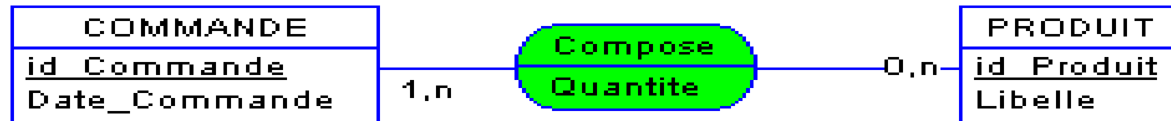


REGLES DE PASSAGE

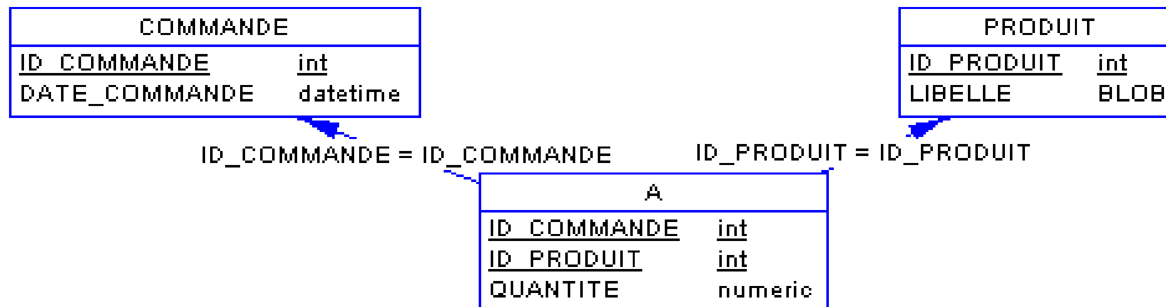
- **3 : Relation binaire aux cardinalités (X,n) - (X,n), X=0 ou X=1**

  Il y a création d'une table supplémentaire ayant comme Clé Primaire une clé composée des identifiants des 2 entités. On dit que la Clé Primaire de la nouvelle table est la concaténation des Clés Primaires des deux autres tables.
  Si la relation est porteuse de donnée, celles ci deviennent des attributs pour la nouvelle table.

- **S.I. :**
  Une commande est composée de 1 ou n produits distincts en certaine quantité. Un produit est présent dans 0 ou n commandes en certaine quantité.
  MCD :



  MLD :
  COMMANDE (id_Commande, Date_commande)
  PRODUIT (id_Produit, libelle)
  COMPOSE (id_Commande, id_Produit, qantité)



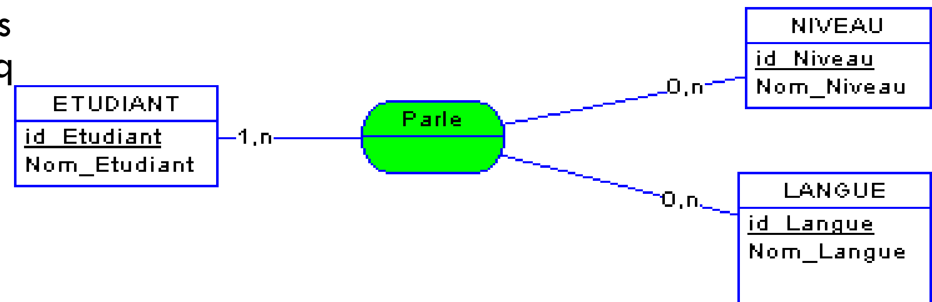
Représentation graphique:

REGLES DE PASSAGE

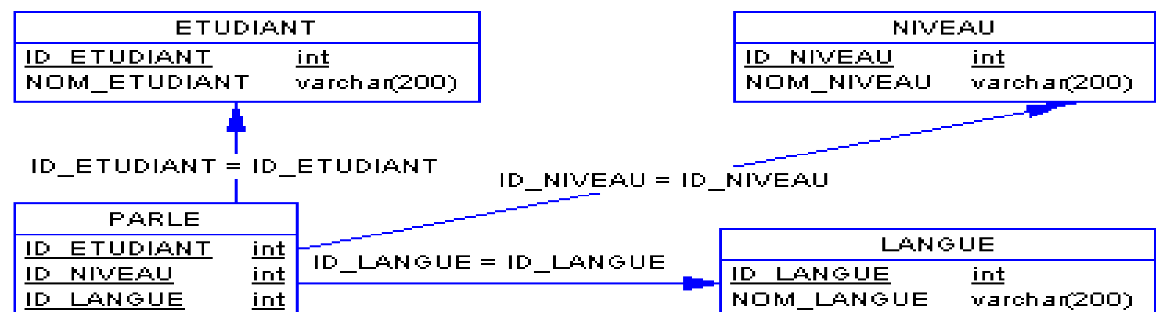
- **4 : Relation n-aire (quelles que soient les cardinalités).**

  Il y a création d'une table supplémentaire ayant comme **Clé Primaire** la **concaténation** des **identifiants** des entités participant à la relation.
  Si la relation est porteuse de donnée, celles ci deviennent des attributs pour la nouvelle table.

- S.I. :
  Un étudiant parle une ou plusieurs langues
  0 ou n étudiants avec un niveau. Pour chaq
  une langue.
  MCD :



MLDR :
ETUDIANT (id_Etudiant, Nom_Etudiant)
NIVEAU (id_Niveau, Nom_Niveau)
LANGUE (id_Langue, Nom_Langue)
PARLE (id_Etudiant, id_Niveau, id_Langue)
Graphiquement:



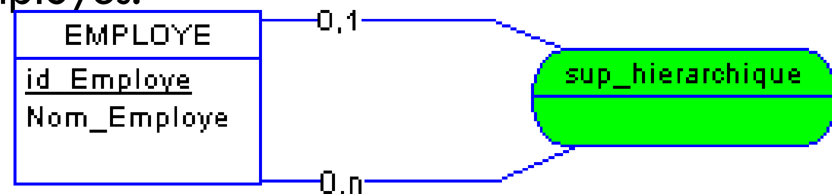
REGLES DE PASSAGE

- **5 : Association Réflexive.**
  Premier cas : cardinalité (X,1) - (X,n), avec X=0 ou X=1.

  La **Clé Primaire** de l'entité se dédouble et devient une **Clé Etrangère** dans la relation ou nouvelle table. Exactement comme si l'entité se dédoublait et était reliée par une relation binaire (X,1) - (X,n)

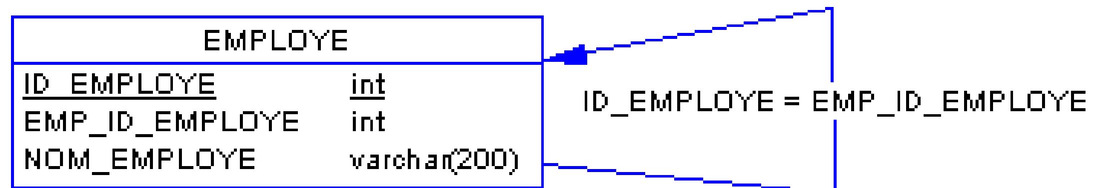
- S.I. :
  Prenons l'exemple d'une société organisée de manière pyramidale : chaque employé a 0 ou 1 supérieur hiérarchique direct. Simultanément, chaque employé est le supérieur hiérarchique direct de 0 ou plusieurs employés.

  MCD :



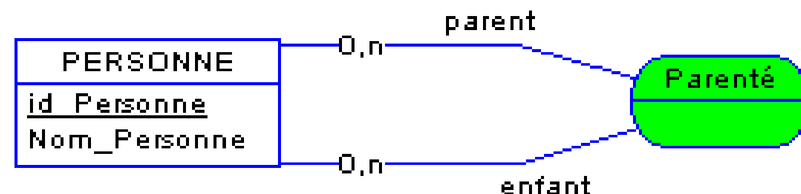
  MLDR :

  EMPLOYE (id_Employe, Nom_Employe, #id_Sup_Hierarchique)
  #id_Sup_Hierarchique est l'identifiant (id_Employe) du supérieur hiérarchique direct de l'employé considéré.



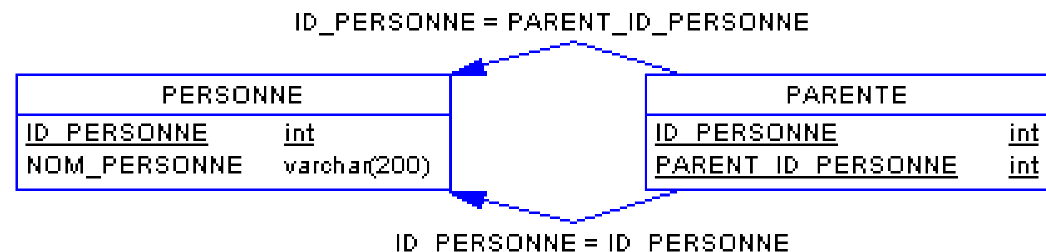
REGLES DE PASSAGE

- Deuxième cas : cardinalité (X,n) - (X,n), avec X=0 ou X=1.
  De même, tout se passe exactement comme si l'entité se dédoublait et était reliée par une relation binaire (X,n) - (X,n) (Cf règle 3). Il y a donc création d'une nouvelle table.
- S.I. :
  Prenons cette fois l'exemple d'une organisation de type familiale : chaque personne a 0 ou n descendants directs (enfants), et a aussi 0 ou n descendants directs (enfants).
  MCD :



MLDR :

PERSONNE (id_Personne, Nom_Personne)
PARENTE (#id_Parent, #id_Enfant)
#id_Parent est l'identifiant (id_Personne) d'un ascendant direct de la personne.
#id_Enfant est l'identifiant (id_Personne) d'un descendant direct de la personne.
La table PARENTE sera en fait l'ensemble des couples (parents-enfants) présent dans cette famille.



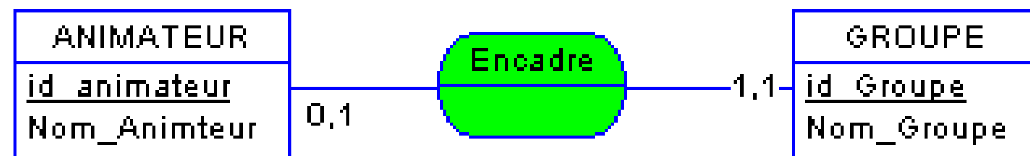
REGLES DE PASSAGE

- **6 : Relation binaire aux cardinalités (0,1) - (1,1).**

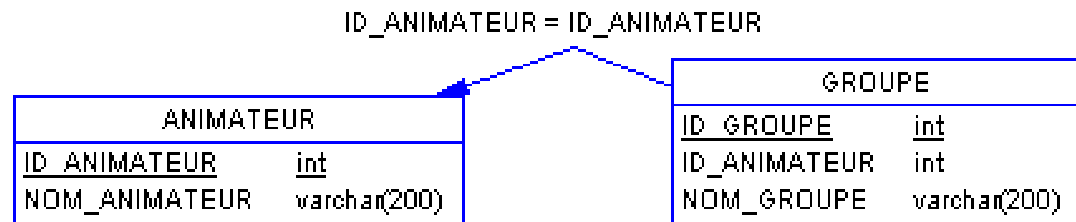
  La **Clé Primaire** de la table à la cardinalité (0,1) devient une **Clé Etrangère** dans la table à la cardinalité (1,1) avec une contrainte d'unicité:

- S.I. :

  Dans ce centre de vacances, Chaque animateur encadre en solo 0 ou 1 groupe, chaque groupe étant encadré par un et un seul animateur.
  MCD :



- MLDR :
  ANIMATEUR (id_Animateur, Nom_Animateur)
  GROUPE (id_Groupe, Nom_Groupe, #id_animateur(non vide,unique))
  Graphiquement:



CONCLUSION

- **Ces 6 règles représentent TOUS les cas de passage du MCD aux MLD que vous pourrez rencontrer. Et surtout, votre base de donnée devra correspondre EXACTEMENT au système d'information décris dans le cahier des charges.**

- **De plus, écrire le MCD, le valider avec votre client, puis en déduire le MLDR qui sera une assise pour la création de la base de données.**

- **la majorité du travail restant ne sera plus qu'une question de requêtes, de mise en forme et d'ergonomie, avec une bonne gestion d'Entrée/Sortie de l'information...**